Configuration Manual

MSc Research Project
Data Analytics

Vasit Ali
Student ID: x22144170

School of Computing
National College of Ireland

Supervisor: Abid Yaqoob

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Vasit Ali
Student ID:	x22144170
Programme:	Data Analytics
Year:	2023
Module:	MSc Research Project
Supervisor:	Abid Yaqoob
Submission Due Date:	31/01/2024
Project Title:	Configuration Manual
Word Count:	618
Page Count:	24

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Vasit Ali
Date:	31st January 2024

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Vasit Ali
x22144170

1 Introduction

This manual illustrates how to execute and configure the implementation code for the current research project. This document provides specified details about the machine hardware as well as the programs to run. Following the below steps will enable the users to generate summaries of the research papers.

2 System Specification

2.1 Hardware Specification

Following are the hardware specifications of the system that was used to develop the project:

Component	**Specifications**
Processor	12th Generation Intel® Core™ i9-12900H processor
RAM	16 Gb/s, NVMe
Storage	1 TB SSD, PCIe Gen4
Graphics Card	NVIDIA® GeForce RTX 3060 with 6 GB of dedicated GDDR6 VRAM
Operating System	Windows 11 Home 64-bit

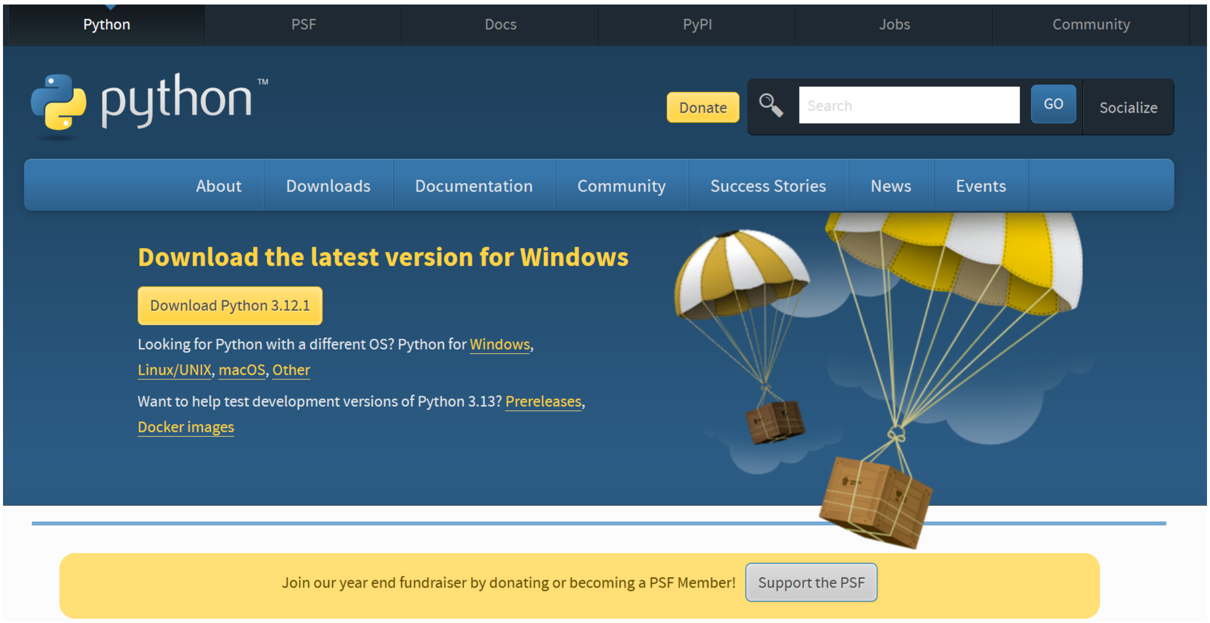
Table 1: Hardware Specifications

2.2 Software Specification

Following are the software specifications of the system that was used to develop the project:

Software	**Specifications**
Operating System	Windows 11 Home 64-bit
IDE	Jupyter Notebook
Scripting Language	Python 3.7

Table 2: Software Specifications



Download the latest version for Windows

Looking for Python with a different OS? Python for Windows, Linux/UNIX, macOS, Other

Want to help test development versions of Python 3.13? Prereleases, Docker images

Join our year end fundraiser by donating or becoming a PSF Member!

Figure 1: Python's Official Website Page

3 Software Tools

Following are the software tools that were used to implement the project.

3.1 Python

Python programming language was used to develop the project. The main reason to choose Python was its useful libraries for Data cleaning, visualization, and deep learning models. Python was downloaded from the main website[1]. Figure 1 shows the download page of Python's official website.

3.2 Jupyter Notebook

Jupyter Notebook was used as a compiler to run the code as it allows the users to implement all the code in one place and execute the codes in small parts like cells to allow the audience to check the output of each code with ease. Jupyter Notebook was downloaded from its official website[2] and Figure 2 illustrates its download page

4 Packages and Libraries

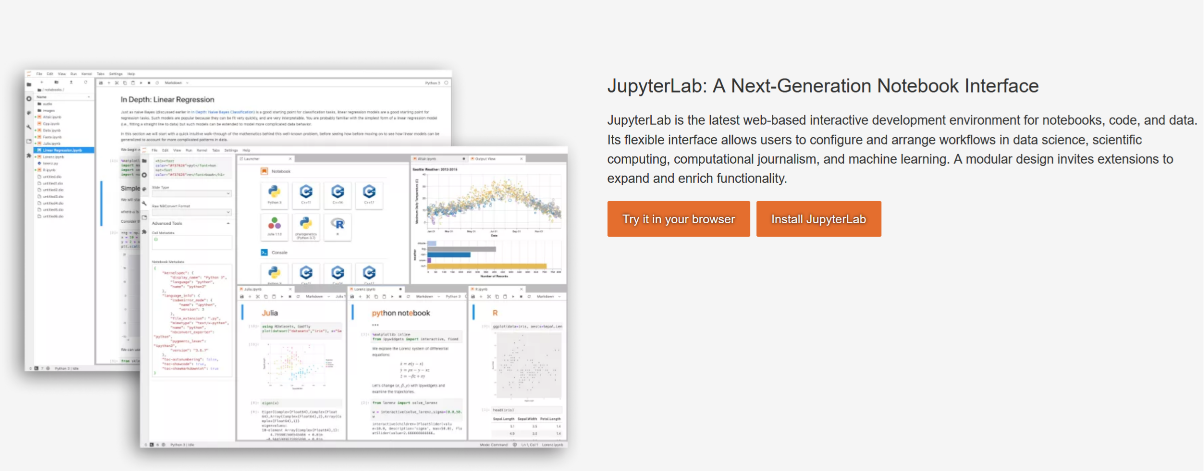
4.1 Python Packages

Following are the Python packages which were installed using pip and used to implement the project as shown in Figure 3 and Figure 4

- **scikit-learn**

[1] https://www.python.org/downloads/

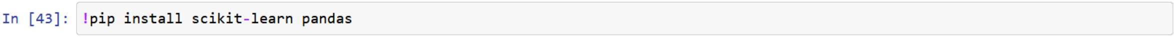
[2] https://jupyter.org/



JupyterLab: A Next-Generation Notebook Interface

JupyterLab is the latest web-based interactive development environment for notebooks, code, and data. Its flexible interface allows users to configure and arrange workflows in data science, scientific computing, computational journalism, and machine learning. A modular design invites extensions to expand and enrich functionality.

Figure 2: Jupyter Notebook's Official Website Page



```
In [43]: !pip install scikit-learn pandas
```

Figure 3: Python Package scikit-learn

- **Keras**
- **tensorflow**

4.2 Python Libraries

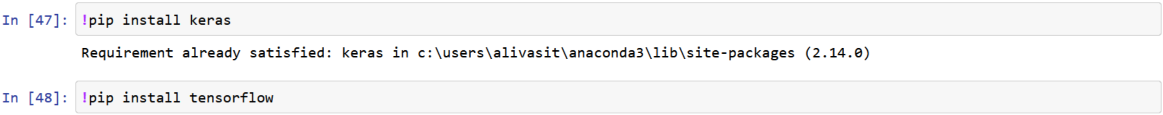
Following are the Python libraries which were installed and used to implement the project as shown in Figure 5

5 Implementation

Pandas library was used to load and check the dataset as can be seen in Figure 6

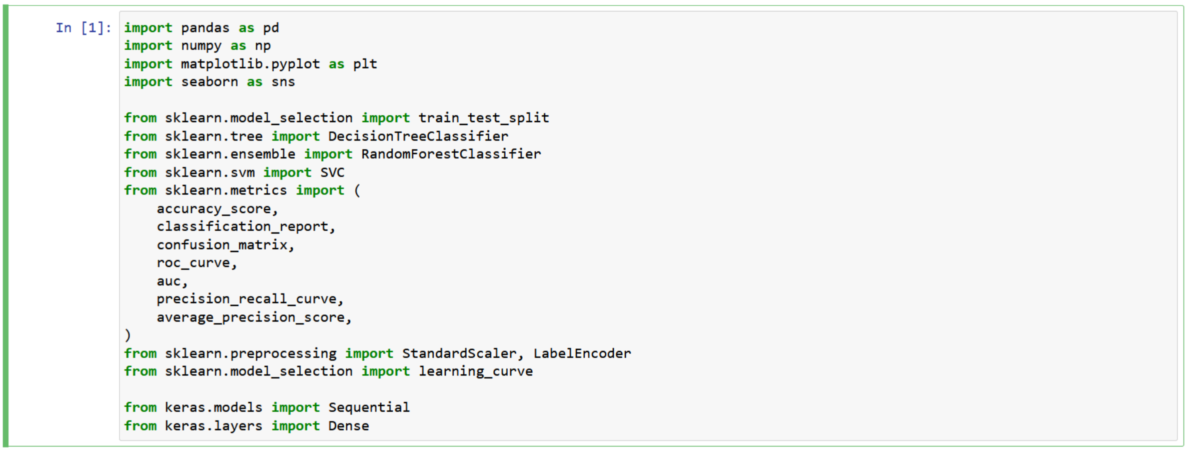
- **Data Cleaning**

References



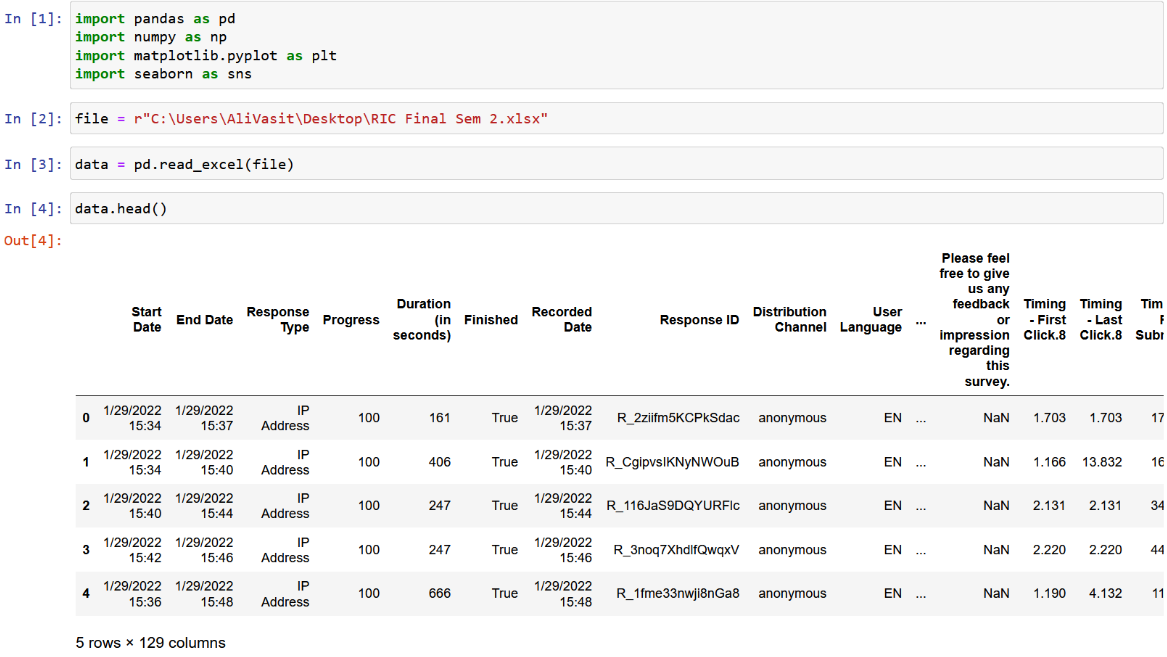
```
In [47]: !pip install keras
         Requirement already satisfied: keras in c:\users\alivasit\anaconda3\lib\site-packages (2.14.0)

In [48]: !pip install tensorflow
```

Figure 4: Python Package Keras and tensorflow



```
In [1]: import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        import seaborn as sns

        from sklearn.model_selection import train_test_split
        from sklearn.tree import DecisionTreeClassifier
        from sklearn.ensemble import RandomForestClassifier
        from sklearn.svm import SVC
        from sklearn.metrics import (
            accuracy_score,
            classification_report,
            confusion_matrix,
            roc_curve,
            auc,
            precision_recall_curve,
            average_precision_score,
        )
        from sklearn.preprocessing import StandardScaler, LabelEncoder
        from sklearn.model_selection import learning_curve

        from keras.models import Sequential
        from keras.layers import Dense
```

Figure 5: Python Libraries

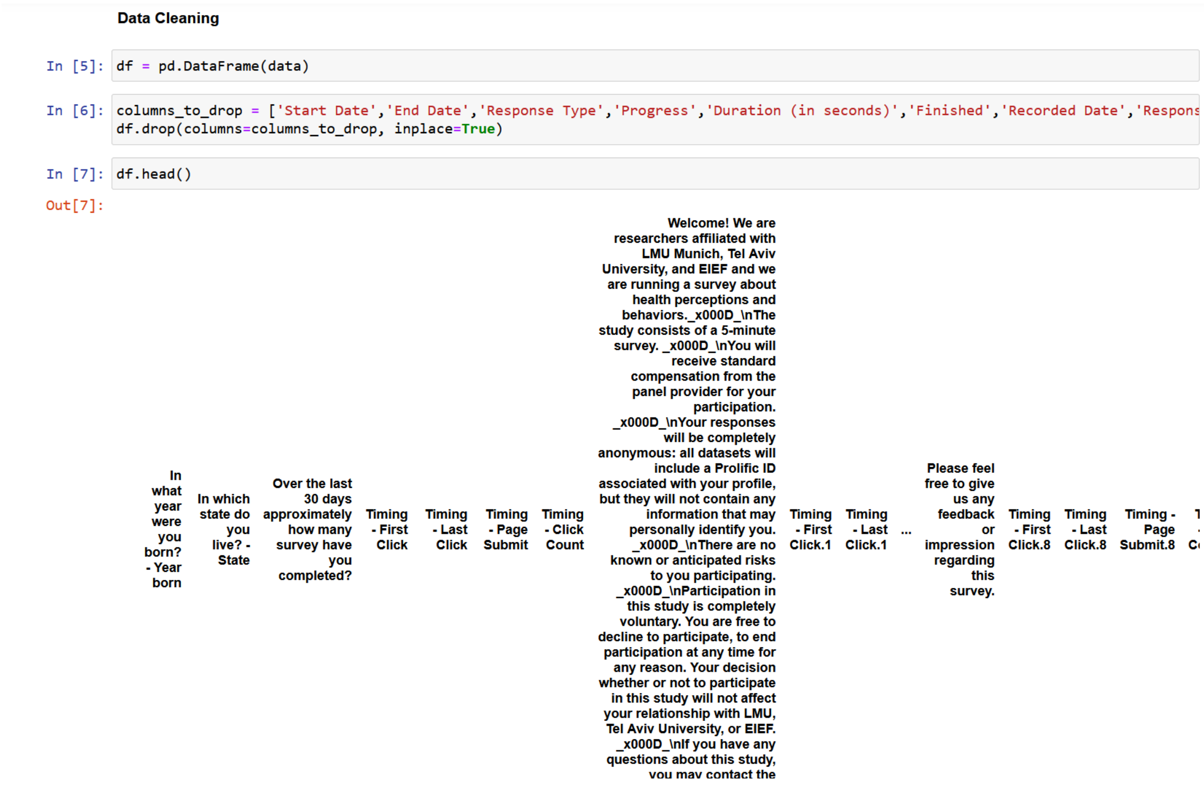


```
In [1]: import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        import seaborn as sns

In [2]: file = r"C:\Users\AliVasit\Desktop\RIC Final Sem 2.xlsx"

In [3]: data = pd.read_excel(file)

In [4]: data.head()

Out[4]:
```

	Start Date	End Date	Response Type	Progress	Duration (in seconds)	Finished	Recorded Date	Response ID	Distribution Channel	User Language	...	Please feel free to give us any feedback or impression regarding this survey.	Timing - First Click.8	Timing - Last Click.8	Tim F Subr
0	1/29/2022 15:34	1/29/2022 15:37	IP Address	100	161	True	1/29/2022 15:37	R_2ziifm5KCPkSdac	anonymous	EN	...	NaN	1.703	1.703	17
1	1/29/2022 15:34	1/29/2022 15:40	IP Address	100	406	True	1/29/2022 15:40	R_CgipvsIKNyNWOuB	anonymous	EN	...	NaN	1.166	13.832	16
2	1/29/2022 15:40	1/29/2022 15:44	IP Address	100	247	True	1/29/2022 15:44	R_116JaS9DQYURFlc	anonymous	EN	...	NaN	2.131	2.131	34
3	1/29/2022 15:42	1/29/2022 15:46	IP Address	100	247	True	1/29/2022 15:46	R_3noq7XhdlfQwqxV	anonymous	EN	...	NaN	2.220	2.220	44
4	1/29/2022 15:36	1/29/2022 15:48	IP Address	100	666	True	1/29/2022 15:48	R_1fme33nwji8nGa8	anonymous	EN	...	NaN	1.190	4.132	11

5 rows × 129 columns

Figure 6: Loaded Dataset

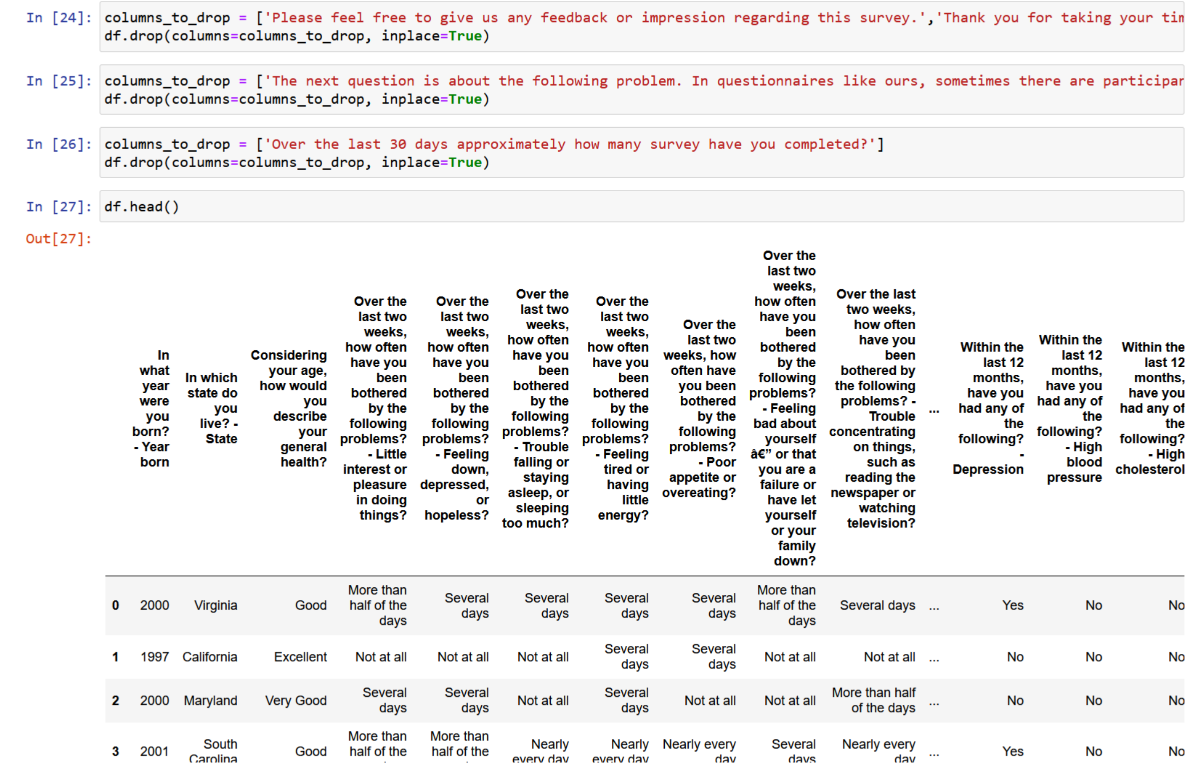


Data Cleaning

```
In [5]: df = pd.DataFrame(data)

In [6]: columns_to_drop = ['Start Date','End Date','Response Type','Progress','Duration (in seconds)','Finished','Recorded Date','Respons
        df.drop(columns=columns_to_drop, inplace=True)

In [7]: df.head()

Out[7]:
```

	In what year were you born? - Year born	In which state do you live? - State	Over the last 30 days approximately how many survey have you completed?	Timing - First Click	Timing - Last Click	Timing - Page Submit	Timing - Click Count	Welcome! We are researchers affiliated with LMU Munich, Tel Aviv University, and EIEF and we are running a survey about health perceptions and behaviors._x000D_\nThe study consists of a 5-minute survey. _x000D_\nYou will receive standard compensation from the panel provider for your participation. _x000D_\nYour responses will be completely anonymous: all datasets will include a Prolific ID associated with your profile, but they will not contain any information that may personally identify you. _x000D_\nThere are no known or anticipated risks to you participating. _x000D_\nParticipation in this study is completely voluntary. You are free to decline to participate, to end participation at any time for any reason. Your decision whether or not to participate in this study will not affect your relationship with LMU, Tel Aviv University, or EIEF. _x000D_\nIf you have any questions about this study, you may contact the	Timing - First Click.1	Timing - Last Click.1	...	Please feel free to give us any feedback or impression regarding this survey.	Timing - First Click.8	Timing - Last Click.8	Timing - Page Submit.8	T - C

Figure 7: Data Cleaning Initialization



```
In [24]: columns_to_drop = ['Please feel free to give us any feedback or impression regarding this survey.','Thank you for taking your tim
         df.drop(columns=columns_to_drop, inplace=True)

In [25]: columns_to_drop = ['The next question is about the following problem. In questionnaires like ours, sometimes there are participar
         df.drop(columns=columns_to_drop, inplace=True)

In [26]: columns_to_drop = ['Over the last 30 days approximately how many survey have you completed?']
         df.drop(columns=columns_to_drop, inplace=True)

In [27]: df.head()

Out[27]:
```

	In what year were you born? - Year born	In which state do you live? - State	Considering your age, how would you describe your general health?	Over the last two weeks, how often have you been bothered by the following problems? - Little interest or pleasure in doing things?	Over the last two weeks, how often have you been bothered by the following problems? - Feeling down, depressed, or hopeless?	Over the last two weeks, how often have you been bothered by the following problems? - Trouble falling or staying asleep, or sleeping too much?	Over the last two weeks, how often have you been bothered by the following problems? - Feeling tired or having little energy?	Over the last two weeks, how often have you been bothered by the following problems? - Poor appetite or overeating?	Over the last two weeks, how often have you been bothered by the following problems? - Feeling bad about yourself â€" or that you are a failure or have let yourself or your family down?	Over the last two weeks, how often have you been bothered by the following problems? - Trouble concentrating on things, such as reading the newspaper or watching television?	...	Within the last 12 months, have you had any of the following? - Depression	Within the last 12 months, have you had any of the following? - High blood pressure	Within the last 12 months, have you had any of the following? - High cholesterol
0	2000	Virginia	Good	More than half of the days	Several days	Several days	Several days	Several days	More than half of the days	Several days	...	Yes	No	No
1	1997	California	Excellent	Not at all	Not at all	Not at all	Several days	Several days	Not at all	Not at all	...	No	No	No
2	2000	Maryland	Very Good	Several days	Several days	Not at all	Several days	Not at all	Not at all	More than half of the days	...	No	No	No
3	2001	South Carolina	Good	More than half of the	More than half of the	Nearly every day	Nearly every day	Nearly every day	Several days	Nearly every day	...	Yes	No	No

Figure 8: Dropping Unnecessary Columns

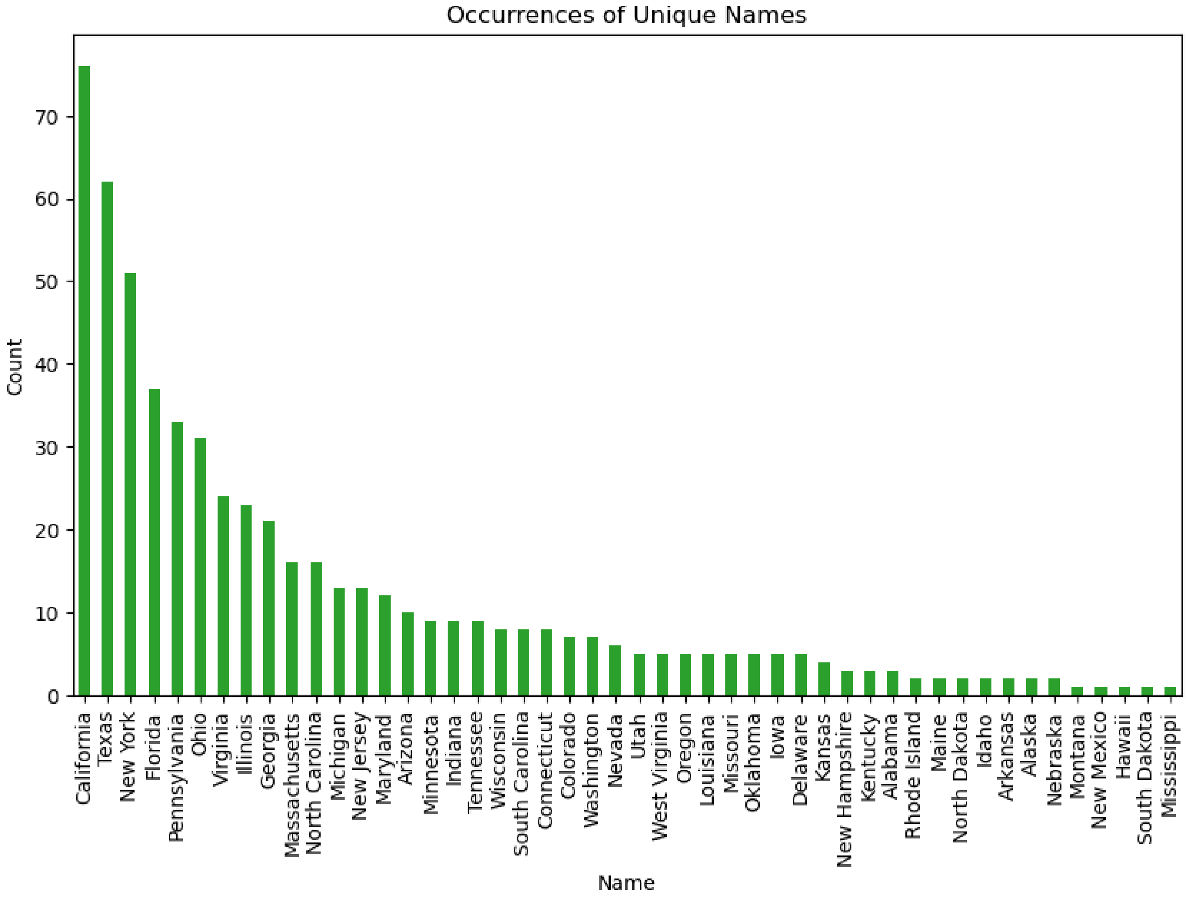
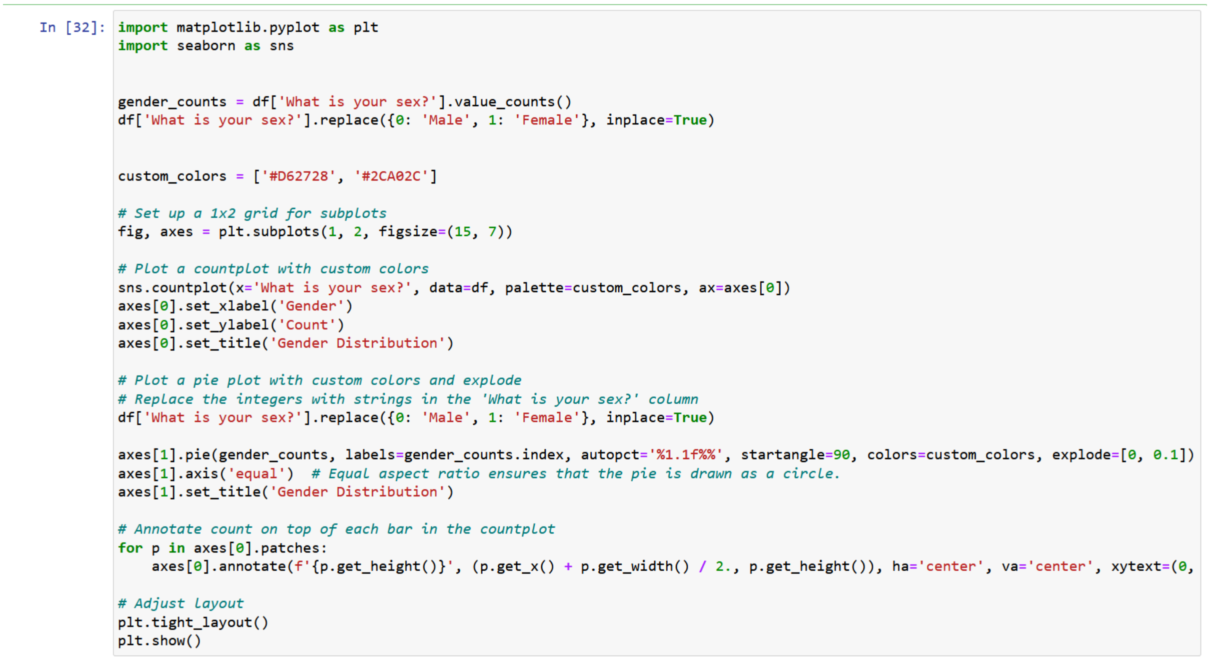


Figure 9: Pre-Visualisation



```
In [32]: import matplotlib.pyplot as plt
         import seaborn as sns


         gender_counts = df['What is your sex?'].value_counts()
         df['What is your sex?'].replace({0: 'Male', 1: 'Female'}, inplace=True)


         custom_colors = ['#D62728', '#2CA02C']

         # Set up a 1x2 grid for subplots
         fig, axes = plt.subplots(1, 2, figsize=(15, 7))

         # Plot a countplot with custom colors
         sns.countplot(x='What is your sex?', data=df, palette=custom_colors, ax=axes[0])
         axes[0].set_xlabel('Gender')
         axes[0].set_ylabel('Count')
         axes[0].set_title('Gender Distribution')

         # Plot a pie plot with custom colors and explode
         # Replace the integers with strings in the 'What is your sex?' column
         df['What is your sex?'].replace({0: 'Male', 1: 'Female'}, inplace=True)

         axes[1].pie(gender_counts, labels=gender_counts.index, autopct='%1.1f%%', startangle=90, colors=custom_colors, explode=[0, 0.1])
         axes[1].axis('equal')  # Equal aspect ratio ensures that the pie is drawn as a circle.
         axes[1].set_title('Gender Distribution')

         # Annotate count on top of each bar in the countplot
         for p in axes[0].patches:
             axes[0].annotate(f'{p.get_height()}', (p.get_x() + p.get_width() / 2., p.get_height()), ha='center', va='center', xytext=(0,

         # Adjust layout
         plt.tight_layout()
         plt.show()
```

Figure 10: Gender Distribution Code

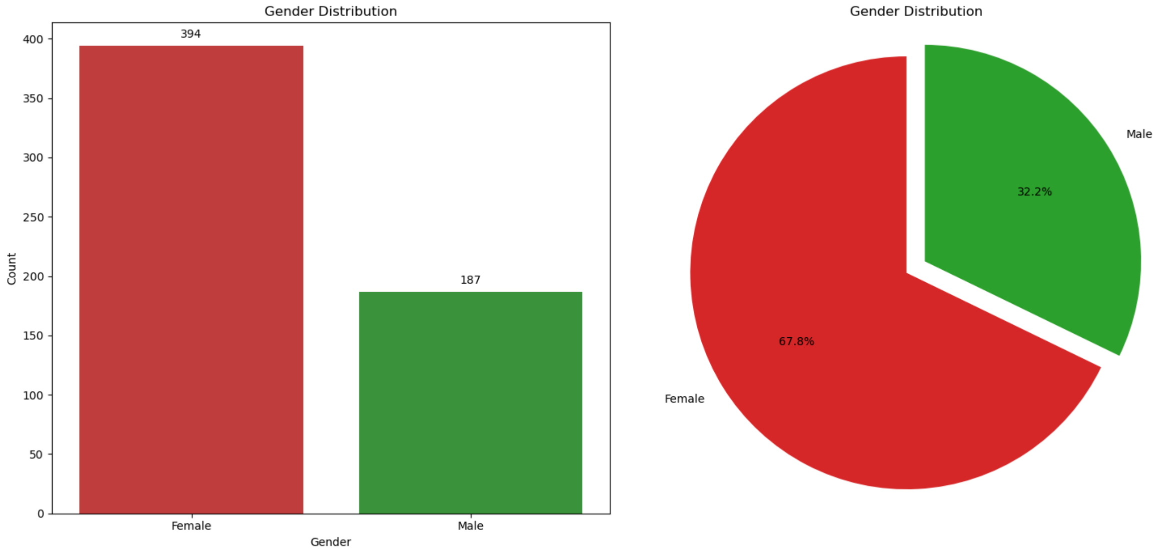
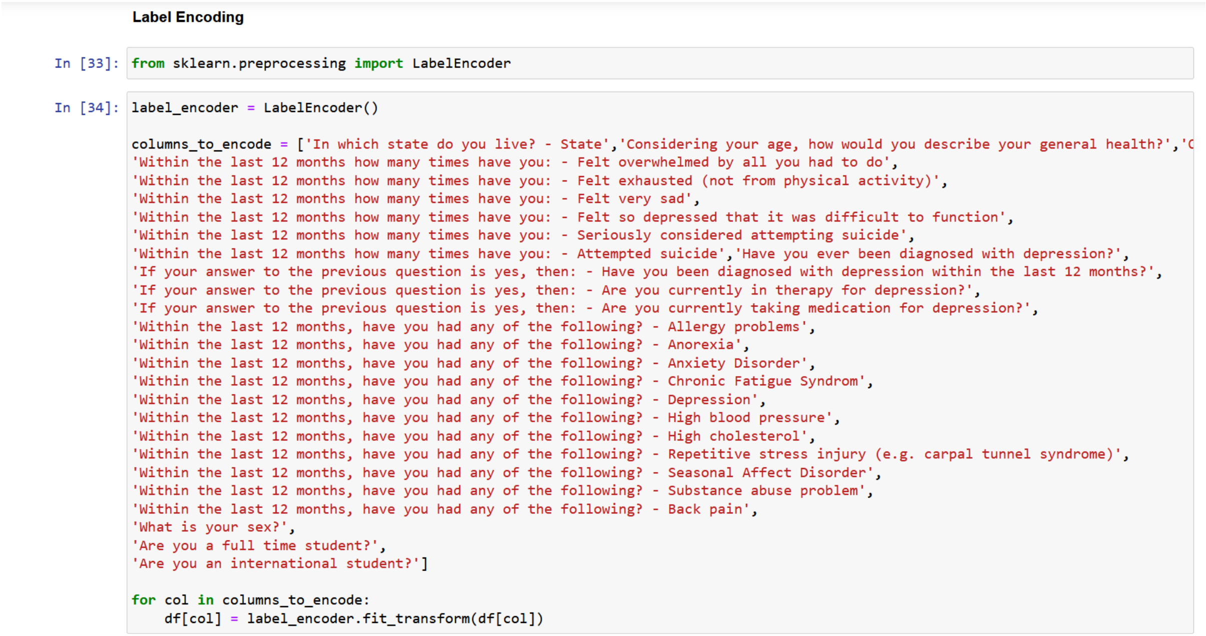


Figure 11: Bar Chart and Pie Plot of Gender Distribution

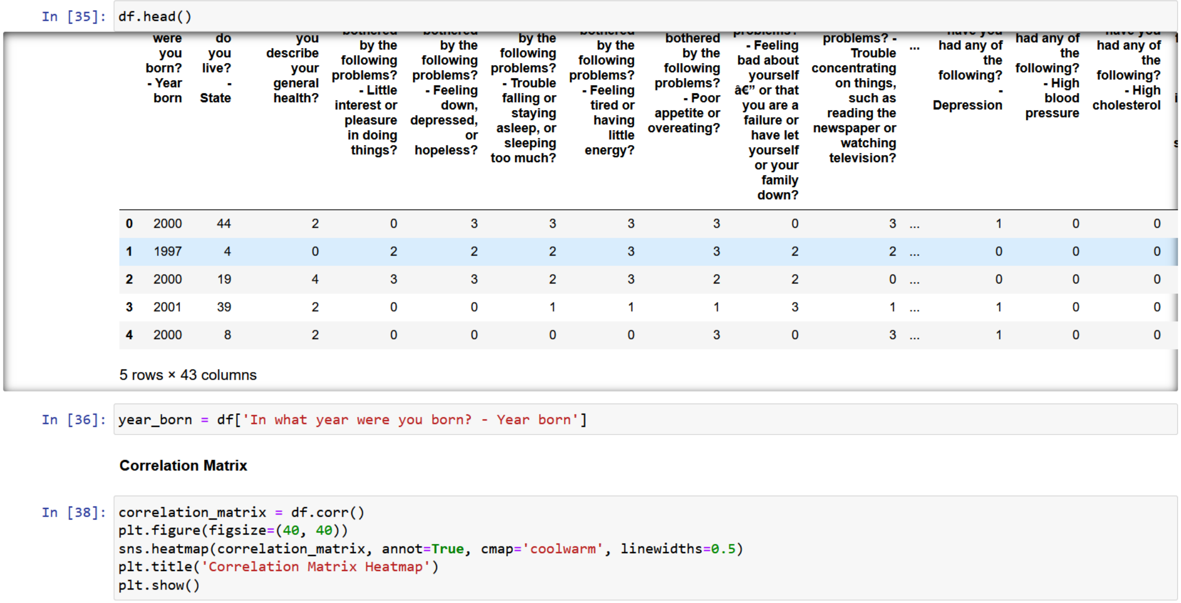


Label Encoding

```
In [33]: from sklearn.preprocessing import LabelEncoder
```

```
In [34]: label_encoder = LabelEncoder()

columns_to_encode = ['In which state do you live? - State','Considering your age, how would you describe your general health?','C
'Within the last 12 months how many times have you: - Felt overwhelmed by all you had to do',
'Within the last 12 months how many times have you: - Felt exhausted (not from physical activity)',
'Within the last 12 months how many times have you: - Felt very sad',
'Within the last 12 months how many times have you: - Felt so depressed that it was difficult to function',
'Within the last 12 months how many times have you: - Seriously considered attempting suicide',
'Within the last 12 months how many times have you: - Attempted suicide','Have you ever been diagnosed with depression?',
'If your answer to the previous question is yes, then: - Have you been diagnosed with depression within the last 12 months?',
'If your answer to the previous question is yes, then: - Are you currently in therapy for depression?',
'If your answer to the previous question is yes, then: - Are you currently taking medication for depression?',
'Within the last 12 months, have you had any of the following? - Allergy problems',
'Within the last 12 months, have you had any of the following? - Anorexia',
'Within the last 12 months, have you had any of the following? - Anxiety Disorder',
'Within the last 12 months, have you had any of the following? - Chronic Fatigue Syndrom',
'Within the last 12 months, have you had any of the following? - Depression',
'Within the last 12 months, have you had any of the following? - High blood pressure',
'Within the last 12 months, have you had any of the following? - High cholesterol',
'Within the last 12 months, have you had any of the following? - Repetitive stress injury (e.g. carpal tunnel syndrome)',
'Within the last 12 months, have you had any of the following? - Seasonal Affect Disorder',
'Within the last 12 months, have you had any of the following? - Substance abuse problem',
'Within the last 12 months, have you had any of the following? - Back pain',
'What is your sex?',
'Are you a full time student?',
'Are you an international student?']

for col in columns_to_encode:
    df[col] = label_encoder.fit_transform(df[col])
```

Figure 12: Label Encoding



```
In [35]: df.head()
```

	were you born? - Year born	do you live? - State	you describe your general health?	bothered by the following problems? - Little interest or pleasure in doing things?	bothered by the following problems? - Feeling down, depressed, or hopeless?	by the following problems? - Trouble falling or staying asleep, or sleeping too much?	bothered by the following problems? - Feeling tired or having little energy?	bothered by the following problems? - Poor appetite or overeating?	problems? - Feeling bad about yourself â€" or that you are a failure or have let yourself or your family down?	problems? - Trouble concentrating on things, such as reading the newspaper or watching television?	...	have you had any of the following? - Depression	had any of the following? - High blood pressure	have you had any of the following? - High cholesterol
0	2000	44	2	0	3	3	3	3	0	3	...	1	0	0
1	1997	4	0	2	2	2	3	3	2	2	...	0	0	0
2	2000	19	4	3	3	2	3	2	2	0	...	0	0	0
3	2001	39	2	0	0	1	1	1	3	1	...	1	0	0
4	2000	8	2	0	0	0	0	3	0	3	...	1	0	0

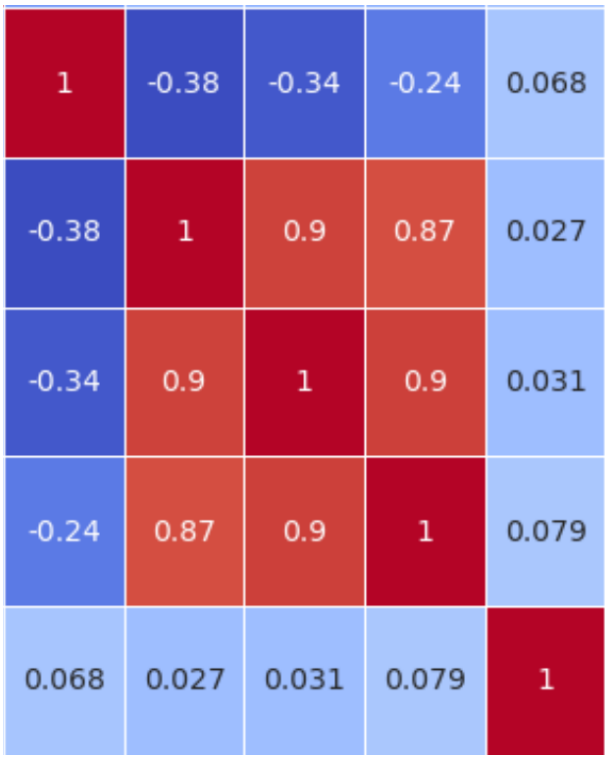
5 rows × 43 columns

```
In [36]: year_born = df['In what year were you born? - Year born']
```

Correlation Matrix

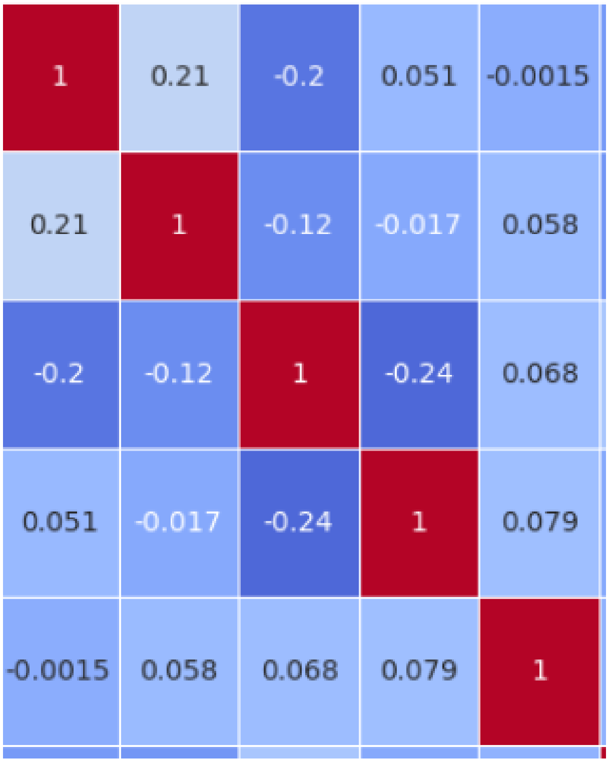
```
In [38]: correlation_matrix = df.corr()
         plt.figure(figsize=(40, 40))
         sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', linewidths=0.5)
         plt.title('Correlation Matrix Heatmap')
         plt.show()
```

Figure 13: Label Encoded data and Correlation Matrix



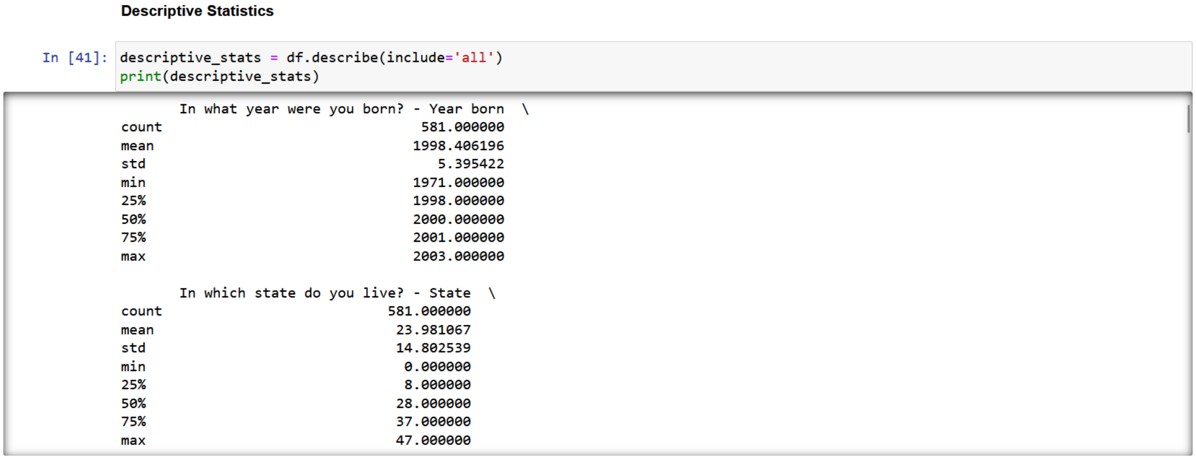
1	-0.38	-0.34	-0.24	0.068
-0.38	1	0.9	0.87	0.027
-0.34	0.9	1	0.9	0.031
-0.24	0.87	0.9	1	0.079
0.068	0.027	0.031	0.079	1

Figure 14: Correlated Variables



1	0.21	-0.2	0.051	-0.0015
0.21	1	-0.12	-0.017	0.058
-0.2	-0.12	1	-0.24	0.068
0.051	-0.017	-0.24	1	0.079
-0.0015	0.058	0.068	0.079	1

Figure 15: Correlation Removed

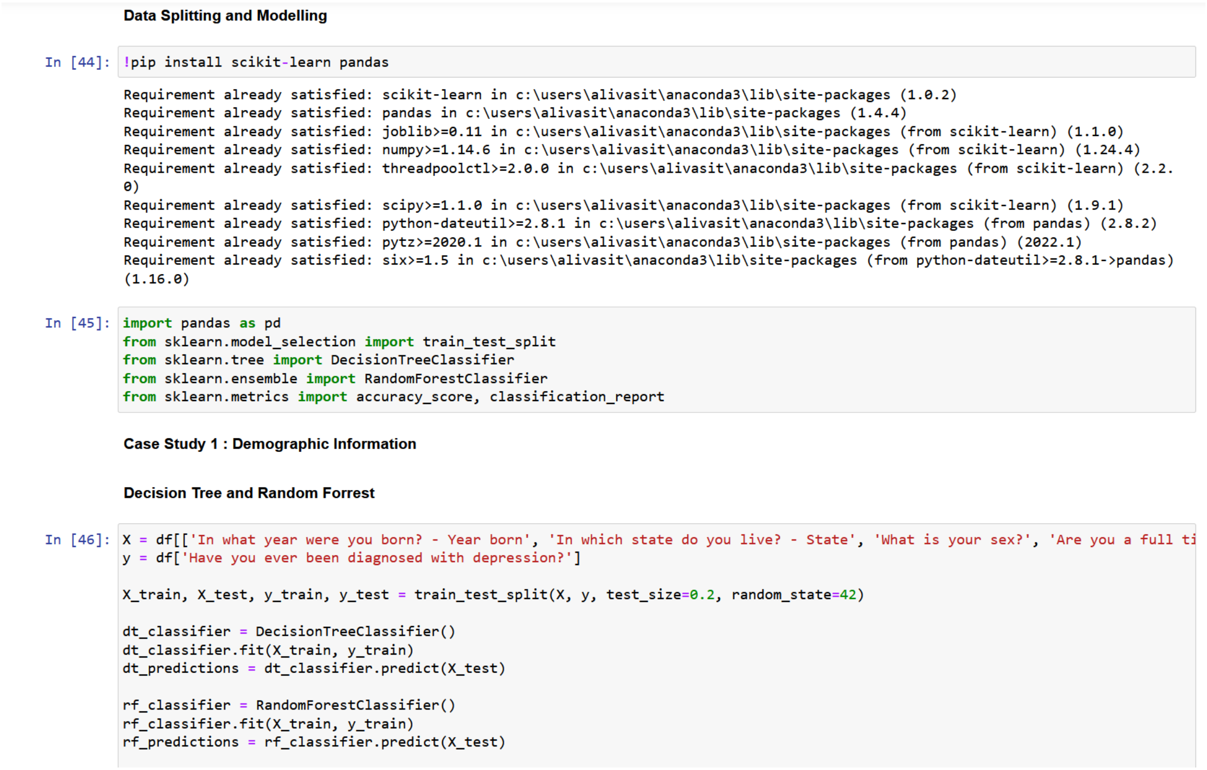


Descriptive Statistics

```
In [41]: descriptive_stats = df.describe(include='all')
         print(descriptive_stats)
```

```
       In what year were you born? - Year born  \
count                               581.000000
mean                               1998.406196
std                                   5.395422
min                                1971.000000
25%                                1998.000000
50%                                2000.000000
75%                                2001.000000
max                                2003.000000

       In which state do you live? - State  \
count                           581.000000
mean                             23.981067
std                              14.802539
min                               0.000000
25%                               8.000000
50%                              28.000000
75%                              37.000000
max                              47.000000
```

Figure 16: Descriptive Statistics



Data Splitting and Modelling

```
In [44]: !pip install scikit-learn pandas
```

```
Requirement already satisfied: scikit-learn in c:\users\alivasit\anaconda3\lib\site-packages (1.0.2)
Requirement already satisfied: pandas in c:\users\alivasit\anaconda3\lib\site-packages (1.4.4)
Requirement already satisfied: joblib>=0.11 in c:\users\alivasit\anaconda3\lib\site-packages (from scikit-learn) (1.1.0)
Requirement already satisfied: numpy>=1.14.6 in c:\users\alivasit\anaconda3\lib\site-packages (from scikit-learn) (1.24.4)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\alivasit\anaconda3\lib\site-packages (from scikit-learn) (2.2.
0)
Requirement already satisfied: scipy>=1.1.0 in c:\users\alivasit\anaconda3\lib\site-packages (from scikit-learn) (1.9.1)
Requirement already satisfied: python-dateutil>=2.8.1 in c:\users\alivasit\anaconda3\lib\site-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\alivasit\anaconda3\lib\site-packages (from pandas) (2022.1)
Requirement already satisfied: six>=1.5 in c:\users\alivasit\anaconda3\lib\site-packages (from python-dateutil>=2.8.1->pandas)
(1.16.0)
```

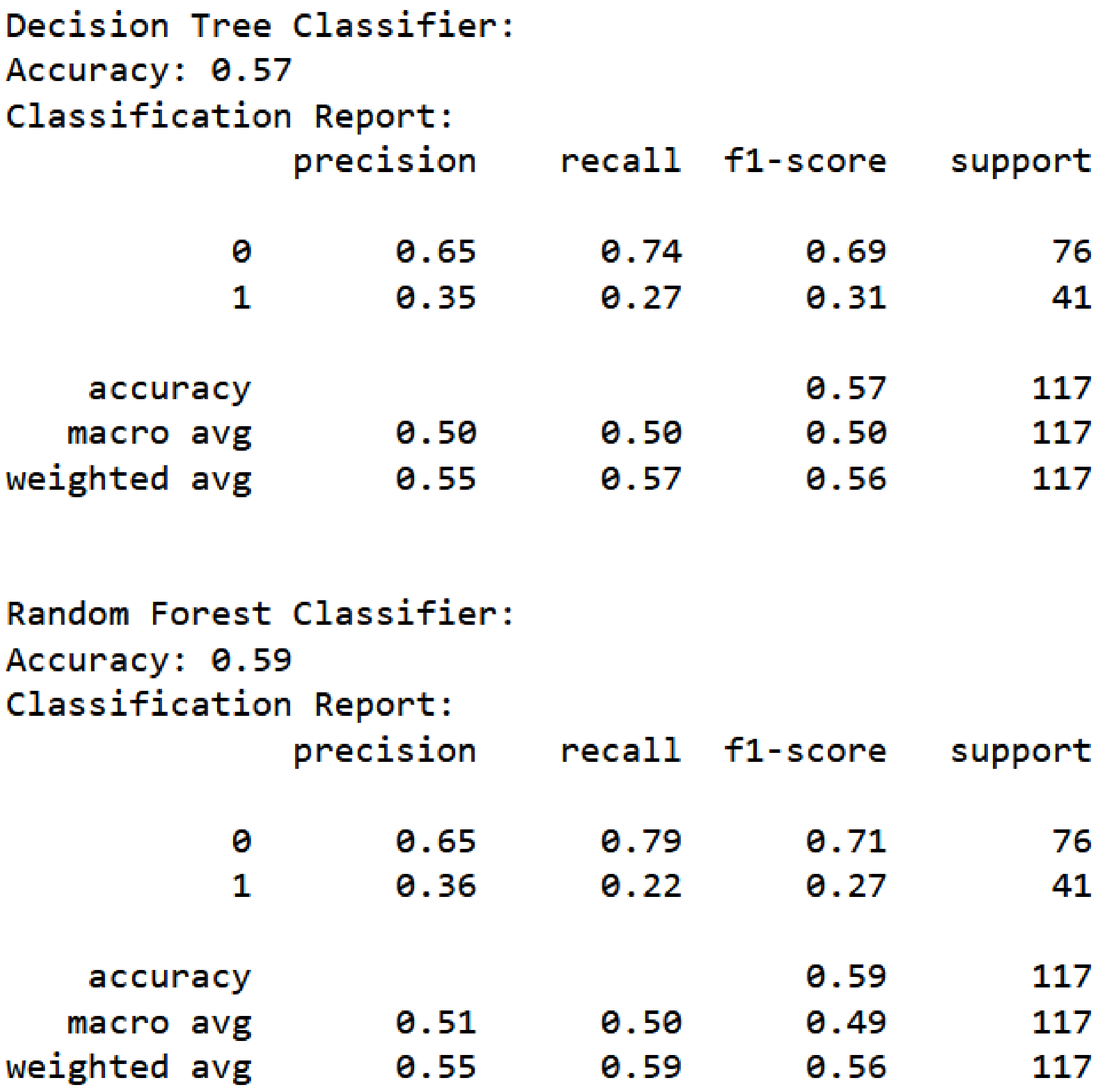
```
In [45]: import pandas as pd
         from sklearn.model_selection import train_test_split
         from sklearn.tree import DecisionTreeClassifier
         from sklearn.ensemble import RandomForestClassifier
         from sklearn.metrics import accuracy_score, classification_report
```

Case Study 1 : Demographic Information

Decision Tree and Random Forrest

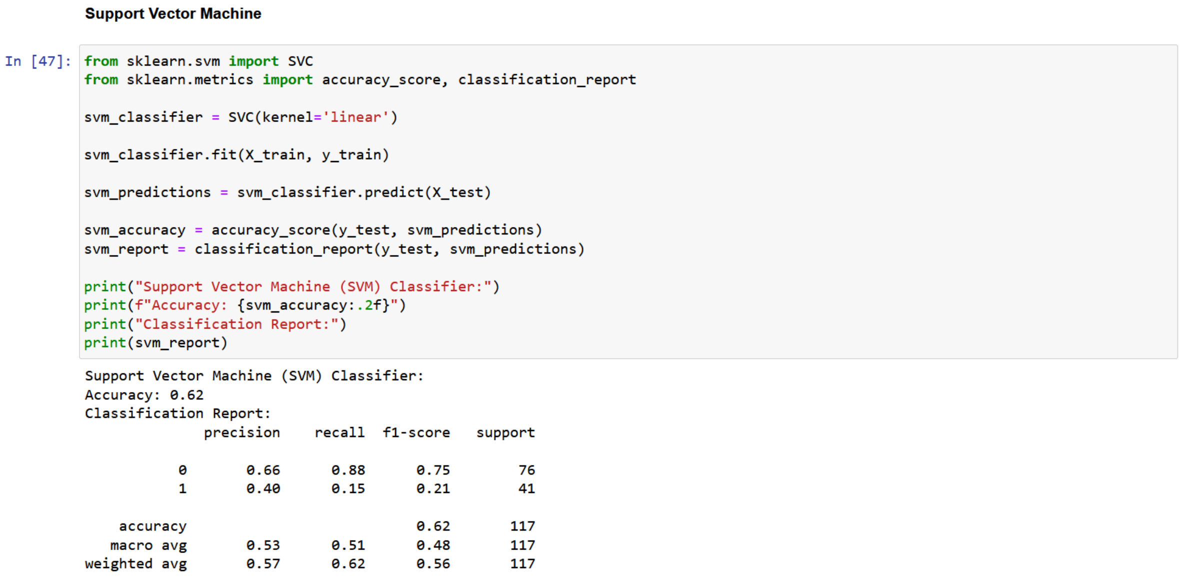
```
In [46]: X = df[['In what year were you born? - Year born', 'In which state do you live? - State', 'What is your sex?', 'Are you a full ti
         y = df['Have you ever been diagnosed with depression?']

         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

         dt_classifier = DecisionTreeClassifier()
         dt_classifier.fit(X_train, y_train)
         dt_predictions = dt_classifier.predict(X_test)

         rf_classifier = RandomForestClassifier()
         rf_classifier.fit(X_train, y_train)
         rf_predictions = rf_classifier.predict(X_test)
```

Figure 17: Data Splitting and Case study 1



```
Decision Tree Classifier:
Accuracy: 0.57
Classification Report:
              precision    recall  f1-score   support

           0       0.65      0.74      0.69        76
           1       0.35      0.27      0.31        41

    accuracy                           0.57       117
   macro avg       0.50      0.50      0.50       117
weighted avg       0.55      0.57      0.56       117


Random Forest Classifier:
Accuracy: 0.59
Classification Report:
              precision    recall  f1-score   support

           0       0.65      0.79      0.71        76
           1       0.36      0.22      0.27        41

    accuracy                           0.59       117
   macro avg       0.51      0.50      0.49       117
weighted avg       0.55      0.59      0.56       117
```

Figure 18: Accuracy scores of Decision Tree and Random Forest

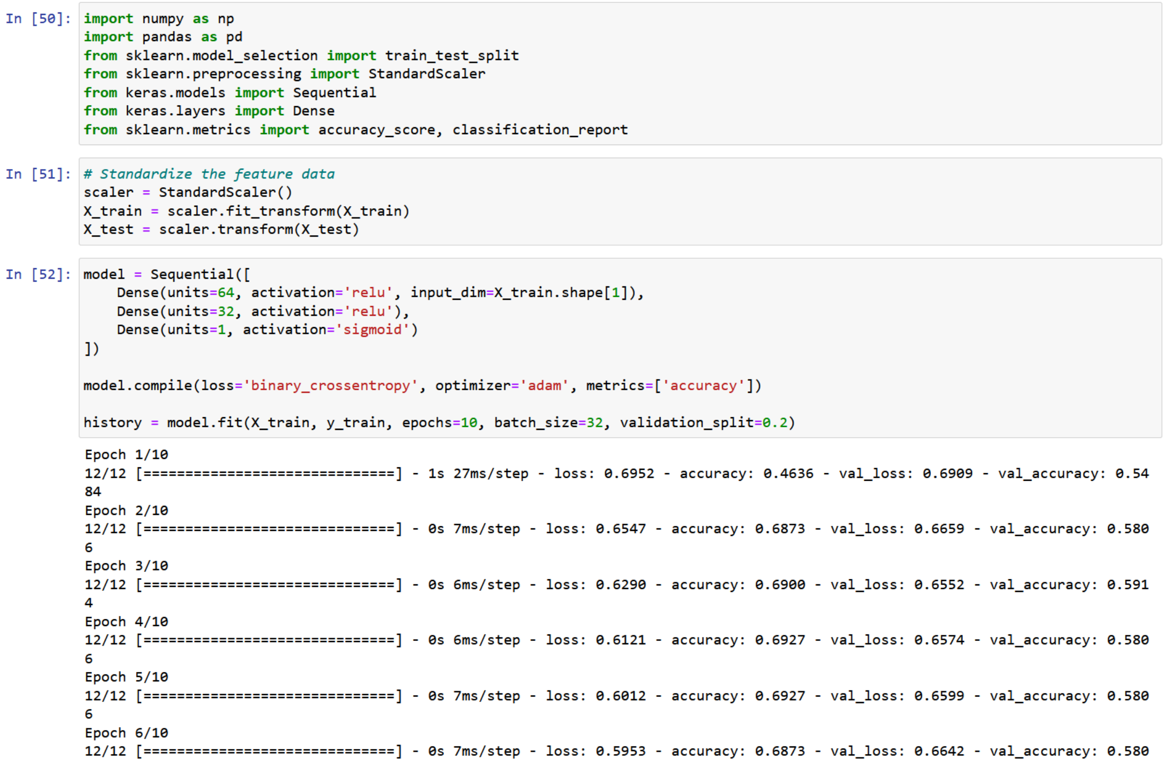


Support Vector Machine

In [47]:
```python
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report

svm_classifier = SVC(kernel='linear')

svm_classifier.fit(X_train, y_train)

svm_predictions = svm_classifier.predict(X_test)

svm_accuracy = accuracy_score(y_test, svm_predictions)
svm_report = classification_report(y_test, svm_predictions)

print("Support Vector Machine (SVM) Classifier:")
print(f"Accuracy: {svm_accuracy:.2f}")
print("Classification Report:")
print(svm_report)
```

```
Support Vector Machine (SVM) Classifier:
Accuracy: 0.62
Classification Report:
              precision    recall  f1-score   support

           0       0.66      0.88      0.75        76
           1       0.40      0.15      0.21        41

    accuracy                           0.62       117
   macro avg       0.53      0.51      0.48       117
weighted avg       0.57      0.62      0.56       117
```

Figure 19: SVM with scores



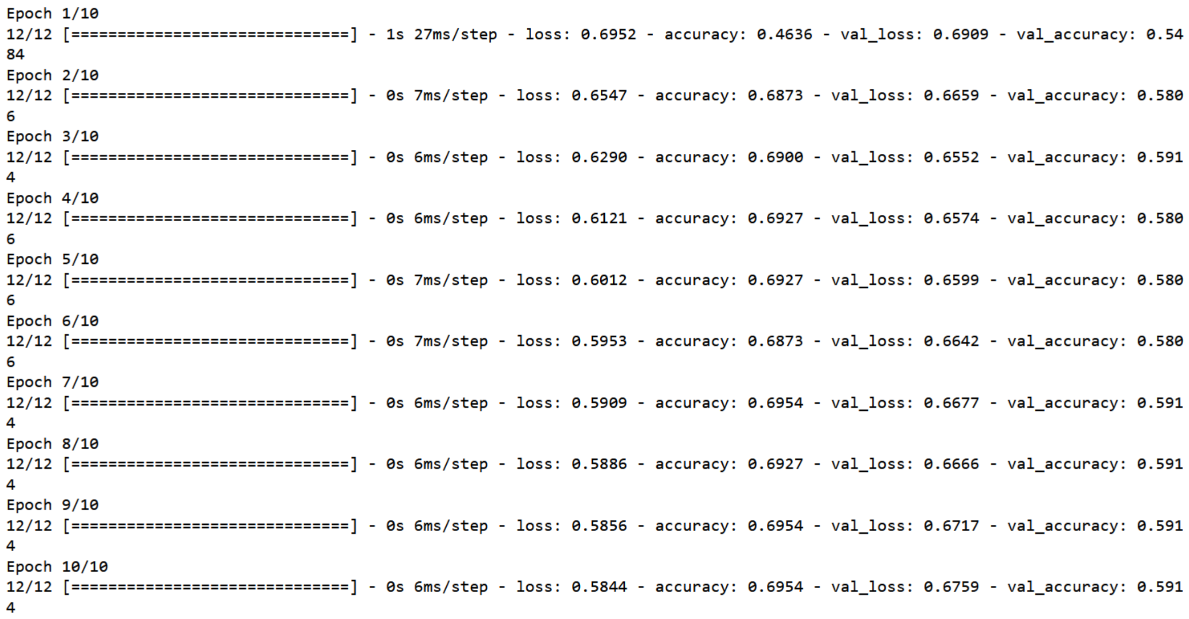
In [50]:
```python
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from keras.models import Sequential
from keras.layers import Dense
from sklearn.metrics import accuracy_score, classification_report
```

In [51]:
```python
# Standardize the feature data
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

In [52]:
```python
model = Sequential([
    Dense(units=64, activation='relu', input_dim=X_train.shape[1]),
    Dense(units=32, activation='relu'),
    Dense(units=1, activation='sigmoid')
])

model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

history = model.fit(X_train, y_train, epochs=10, batch_size=32, validation_split=0.2)
```

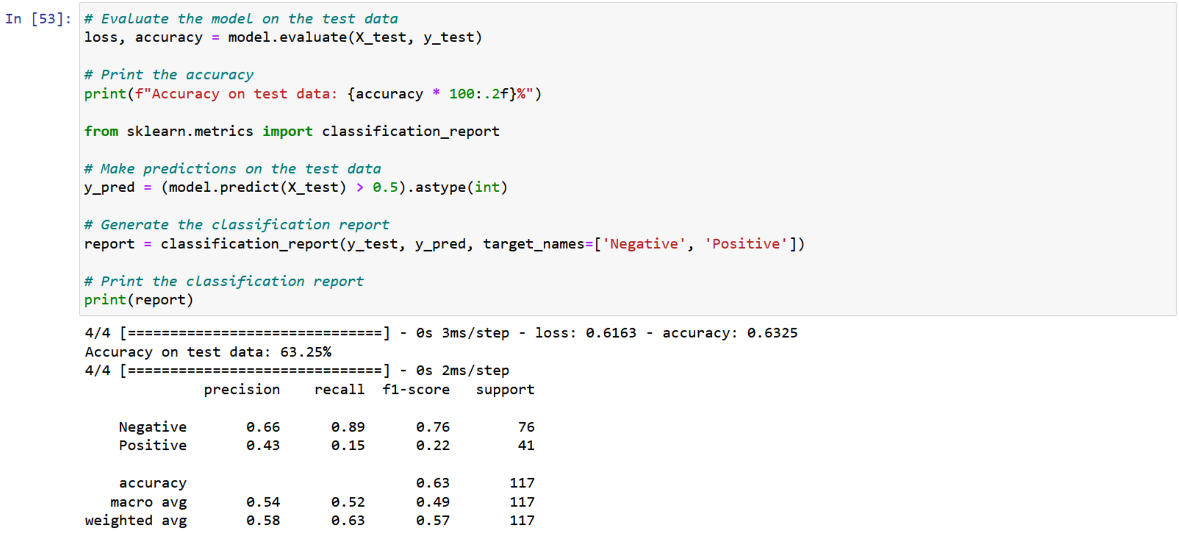
```
Epoch 1/10
12/12 [==============================] - 1s 27ms/step - loss: 0.6952 - accuracy: 0.4636 - val_loss: 0.6909 - val_accuracy: 0.54
84
Epoch 2/10
12/12 [==============================] - 0s 7ms/step - loss: 0.6547 - accuracy: 0.6873 - val_loss: 0.6659 - val_accuracy: 0.580
6
Epoch 3/10
12/12 [==============================] - 0s 6ms/step - loss: 0.6290 - accuracy: 0.6900 - val_loss: 0.6552 - val_accuracy: 0.591
4
Epoch 4/10
12/12 [==============================] - 0s 6ms/step - loss: 0.6121 - accuracy: 0.6927 - val_loss: 0.6574 - val_accuracy: 0.580
6
Epoch 5/10
12/12 [==============================] - 0s 7ms/step - loss: 0.6012 - accuracy: 0.6927 - val_loss: 0.6599 - val_accuracy: 0.580
6
Epoch 6/10
12/12 [==============================] - 0s 7ms/step - loss: 0.5953 - accuracy: 0.6873 - val_loss: 0.6642 - val_accuracy: 0.580
```

Figure 20: Convolutional Neural Network (CNN)



```
Epoch 1/10
12/12 [==============================] - 1s 27ms/step - loss: 0.6952 - accuracy: 0.4636 - val_loss: 0.6909 - val_accuracy: 0.54
84
Epoch 2/10
12/12 [==============================] - 0s 7ms/step - loss: 0.6547 - accuracy: 0.6873 - val_loss: 0.6659 - val_accuracy: 0.580
6
Epoch 3/10
12/12 [==============================] - 0s 6ms/step - loss: 0.6290 - accuracy: 0.6900 - val_loss: 0.6552 - val_accuracy: 0.591
4
Epoch 4/10
12/12 [==============================] - 0s 6ms/step - loss: 0.6121 - accuracy: 0.6927 - val_loss: 0.6574 - val_accuracy: 0.580
6
Epoch 5/10
12/12 [==============================] - 0s 7ms/step - loss: 0.6012 - accuracy: 0.6927 - val_loss: 0.6599 - val_accuracy: 0.580
6
Epoch 6/10
12/12 [==============================] - 0s 7ms/step - loss: 0.5953 - accuracy: 0.6873 - val_loss: 0.6642 - val_accuracy: 0.580
6
Epoch 7/10
12/12 [==============================] - 0s 6ms/step - loss: 0.5909 - accuracy: 0.6954 - val_loss: 0.6677 - val_accuracy: 0.591
4
Epoch 8/10
12/12 [==============================] - 0s 6ms/step - loss: 0.5886 - accuracy: 0.6927 - val_loss: 0.6666 - val_accuracy: 0.591
4
Epoch 9/10
12/12 [==============================] - 0s 6ms/step - loss: 0.5856 - accuracy: 0.6954 - val_loss: 0.6717 - val_accuracy: 0.591
4
Epoch 10/10
12/12 [==============================] - 0s 6ms/step - loss: 0.5844 - accuracy: 0.6954 - val_loss: 0.6759 - val_accuracy: 0.591
4
```

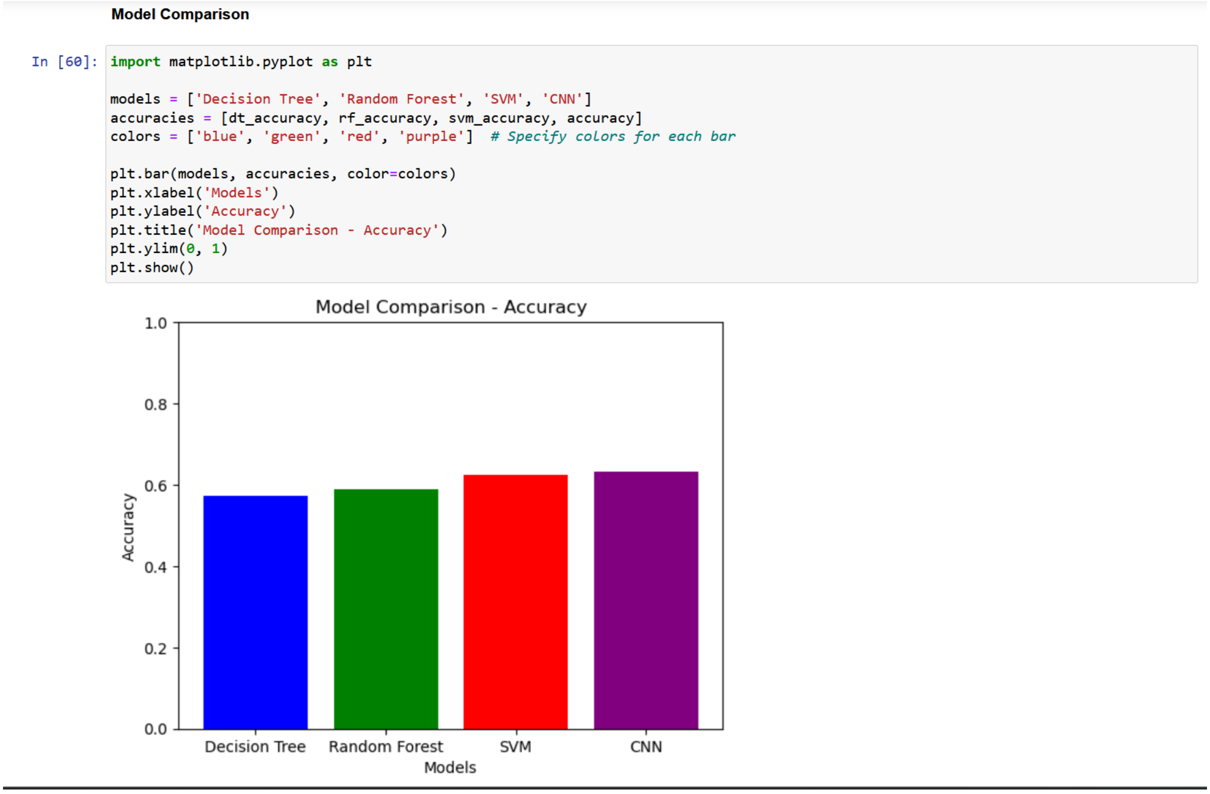
Figure 21: CNN Epochs



```
In [53]: # Evaluate the model on the test data
         loss, accuracy = model.evaluate(X_test, y_test)

         # Print the accuracy
         print(f"Accuracy on test data: {accuracy * 100:.2f}%")

         from sklearn.metrics import classification_report

         # Make predictions on the test data
         y_pred = (model.predict(X_test) > 0.5).astype(int)

         # Generate the classification report
         report = classification_report(y_test, y_pred, target_names=['Negative', 'Positive'])

         # Print the classification report
         print(report)
```

```
4/4 [==============================] - 0s 3ms/step - loss: 0.6163 - accuracy: 0.6325
Accuracy on test data: 63.25%
4/4 [==============================] - 0s 2ms/step
              precision    recall  f1-score   support

    Negative       0.66      0.89      0.76        76
    Positive       0.43      0.15      0.22        41

    accuracy                           0.63       117
   macro avg       0.54      0.52      0.49       117
weighted avg       0.58      0.63      0.57       117
```

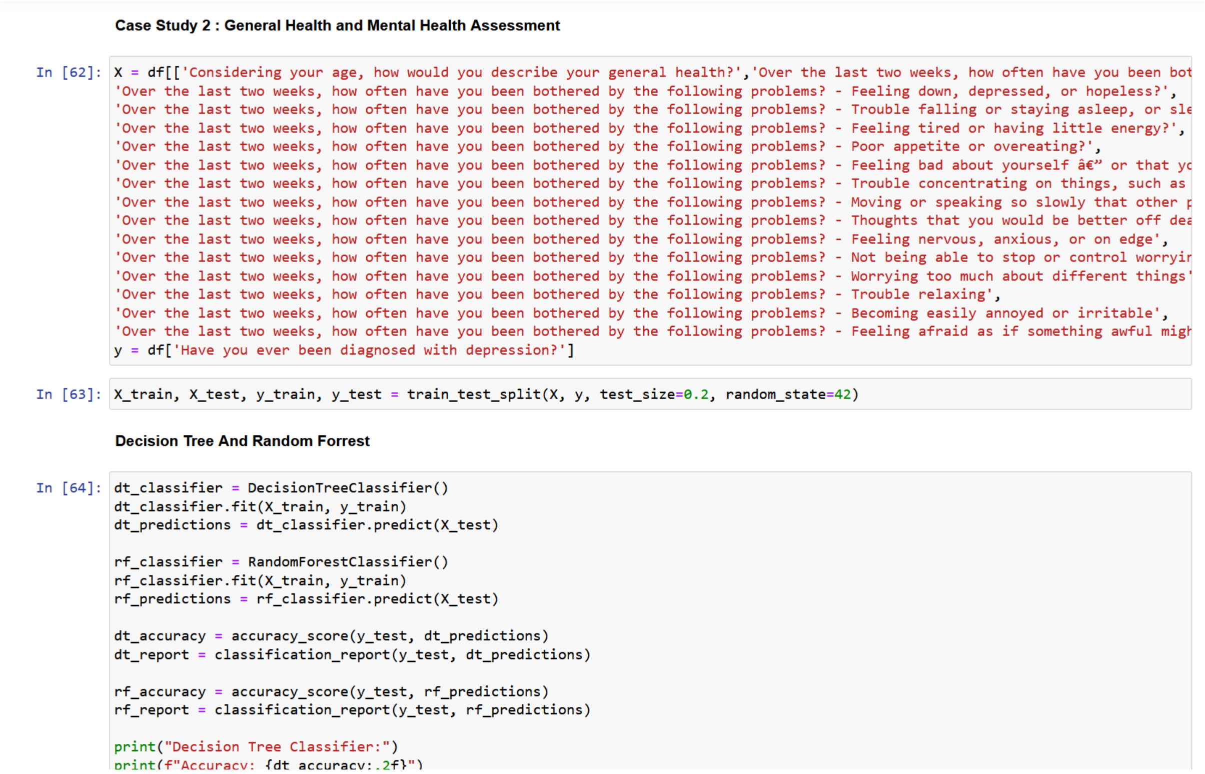
Figure 22: CNN scores



Model Comparison

```
In [60]: import matplotlib.pyplot as plt

         models = ['Decision Tree', 'Random Forest', 'SVM', 'CNN']
         accuracies = [dt_accuracy, rf_accuracy, svm_accuracy, accuracy]
         colors = ['blue', 'green', 'red', 'purple']  # Specify colors for each bar

         plt.bar(models, accuracies, color=colors)
         plt.xlabel('Models')
         plt.ylabel('Accuracy')
         plt.title('Model Comparison - Accuracy')
         plt.ylim(0, 1)
         plt.show()
```

Figure 23: Model Comparison Case Study 1



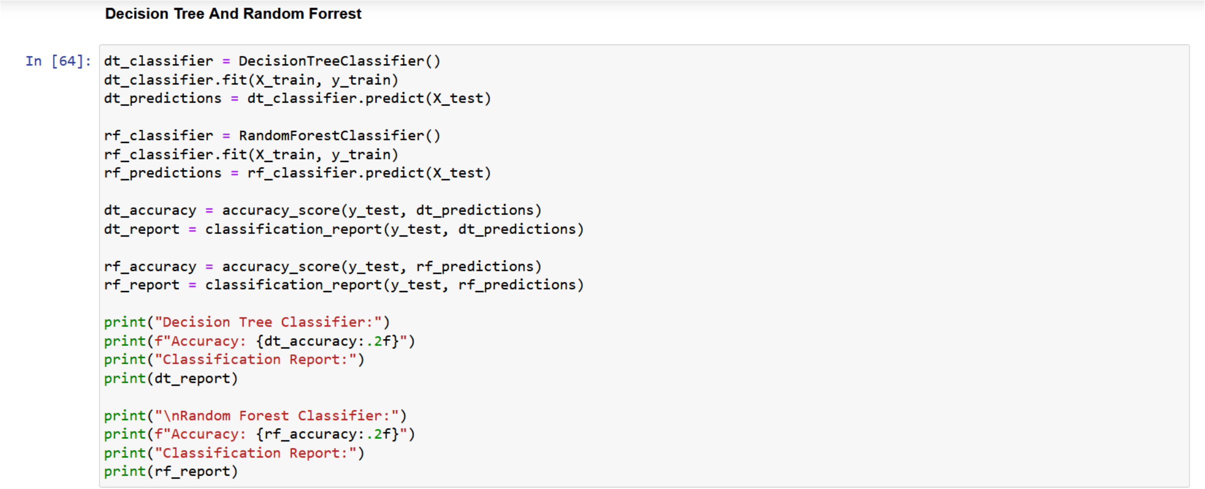
Case Study 2 : General Health and Mental Health Assessment

```
In [62]: X = df[['Considering your age, how would you describe your general health?','Over the last two weeks, how often have you been bot
         'Over the last two weeks, how often have you been bothered by the following problems? - Feeling down, depressed, or hopeless?',
         'Over the last two weeks, how often have you been bothered by the following problems? - Trouble falling or staying asleep, or sle
         'Over the last two weeks, how often have you been bothered by the following problems? - Feeling tired or having little energy?',
         'Over the last two weeks, how often have you been bothered by the following problems? - Poor appetite or overeating?',
         'Over the last two weeks, how often have you been bothered by the following problems? - Feeling bad about yourself â€” or that yo
         'Over the last two weeks, how often have you been bothered by the following problems? - Trouble concentrating on things, such as
         'Over the last two weeks, how often have you been bothered by the following problems? - Moving or speaking so slowly that other p
         'Over the last two weeks, how often have you been bothered by the following problems? - Thoughts that you would be better off dea
         'Over the last two weeks, how often have you been bothered by the following problems? - Feeling nervous, anxious, or on edge',
         'Over the last two weeks, how often have you been bothered by the following problems? - Not being able to stop or control worryin
         'Over the last two weeks, how often have you been bothered by the following problems? - Worrying too much about different things'
         'Over the last two weeks, how often have you been bothered by the following problems? - Trouble relaxing',
         'Over the last two weeks, how often have you been bothered by the following problems? - Becoming easily annoyed or irritable',
         'Over the last two weeks, how often have you been bothered by the following problems? - Feeling afraid as if something awful migh
         y = df['Have you ever been diagnosed with depression?']

In [63]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Decision Tree And Random Forrest

```
In [64]: dt_classifier = DecisionTreeClassifier()
         dt_classifier.fit(X_train, y_train)
         dt_predictions = dt_classifier.predict(X_test)

         rf_classifier = RandomForestClassifier()
         rf_classifier.fit(X_train, y_train)
         rf_predictions = rf_classifier.predict(X_test)

         dt_accuracy = accuracy_score(y_test, dt_predictions)
         dt_report = classification_report(y_test, dt_predictions)

         rf_accuracy = accuracy_score(y_test, rf_predictions)
         rf_report = classification_report(y_test, rf_predictions)

         print("Decision Tree Classifier:")
         print(f"Accuracy: {dt_accuracy:.2f}")
```

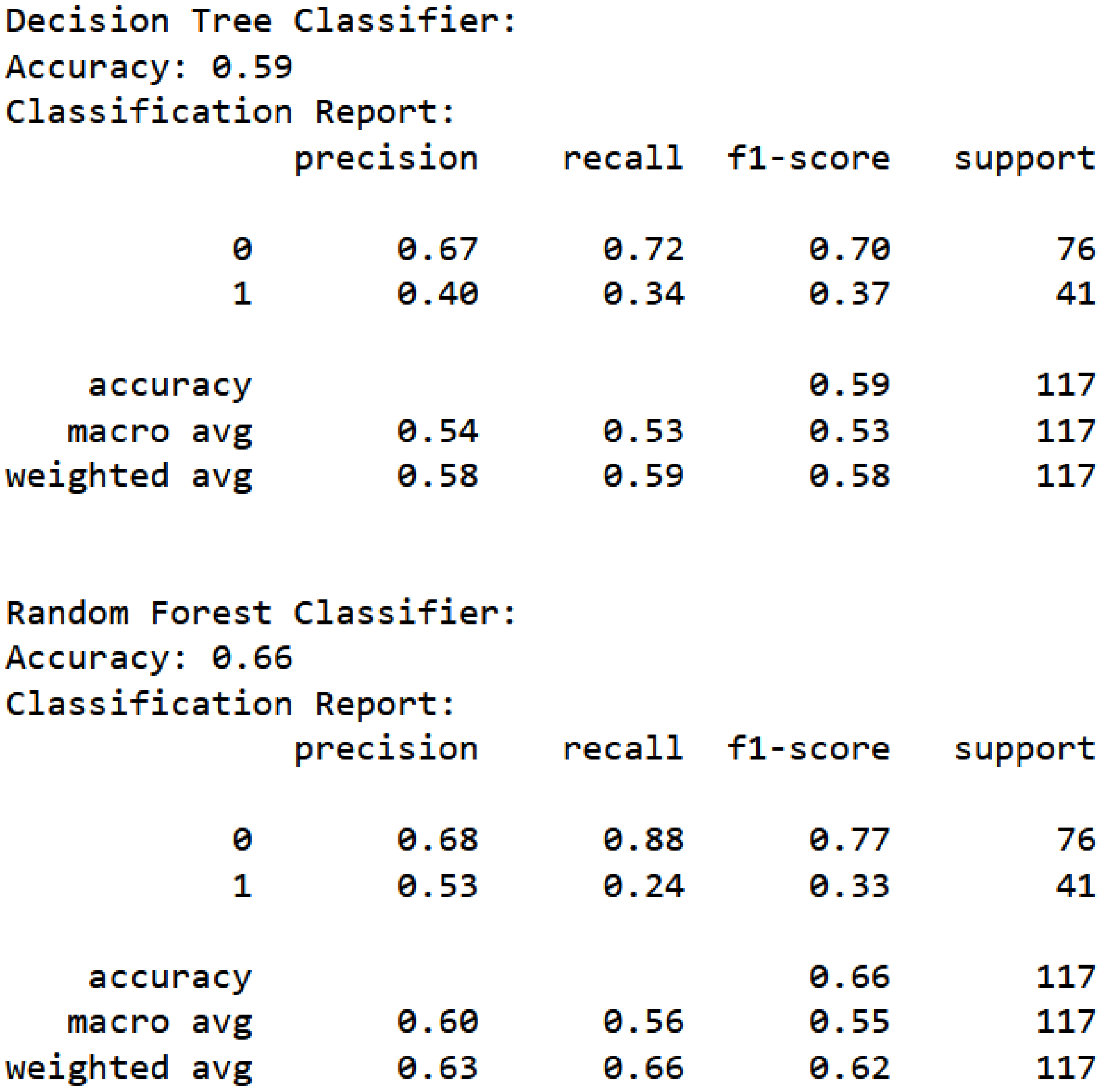
Figure 24: Case Study 2



Decision Tree And Random Forrest

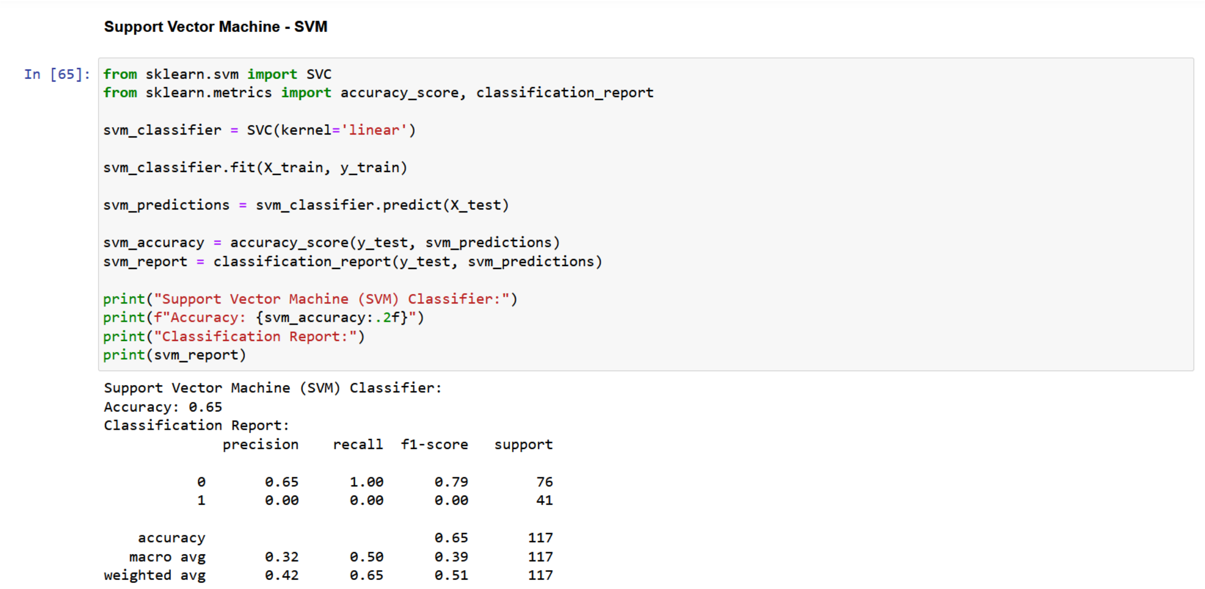
```
In [64]: dt_classifier = DecisionTreeClassifier()
         dt_classifier.fit(X_train, y_train)
         dt_predictions = dt_classifier.predict(X_test)

         rf_classifier = RandomForestClassifier()
         rf_classifier.fit(X_train, y_train)
         rf_predictions = rf_classifier.predict(X_test)

         dt_accuracy = accuracy_score(y_test, dt_predictions)
         dt_report = classification_report(y_test, dt_predictions)

         rf_accuracy = accuracy_score(y_test, rf_predictions)
         rf_report = classification_report(y_test, rf_predictions)

         print("Decision Tree Classifier:")
         print(f"Accuracy: {dt_accuracy:.2f}")
         print("Classification Report:")
         print(dt_report)

         print("\nRandom Forest Classifier:")
         print(f"Accuracy: {rf_accuracy:.2f}")
         print("Classification Report:")
         print(rf_report)
```

Figure 25: Decision Tree and Random Forest Case study 2



```
Decision Tree Classifier:
Accuracy: 0.59
Classification Report:
              precision    recall  f1-score   support

           0       0.67      0.72      0.70        76
           1       0.40      0.34      0.37        41

    accuracy                           0.59       117
   macro avg       0.54      0.53      0.53       117
weighted avg       0.58      0.59      0.58       117


Random Forest Classifier:
Accuracy: 0.66
Classification Report:
              precision    recall  f1-score   support

           0       0.68      0.88      0.77        76
           1       0.53      0.24      0.33        41

    accuracy                           0.66       117
   macro avg       0.60      0.56      0.55       117
weighted avg       0.63      0.66      0.62       117
```

Figure 26: Random Forest and Decision Tree Scores

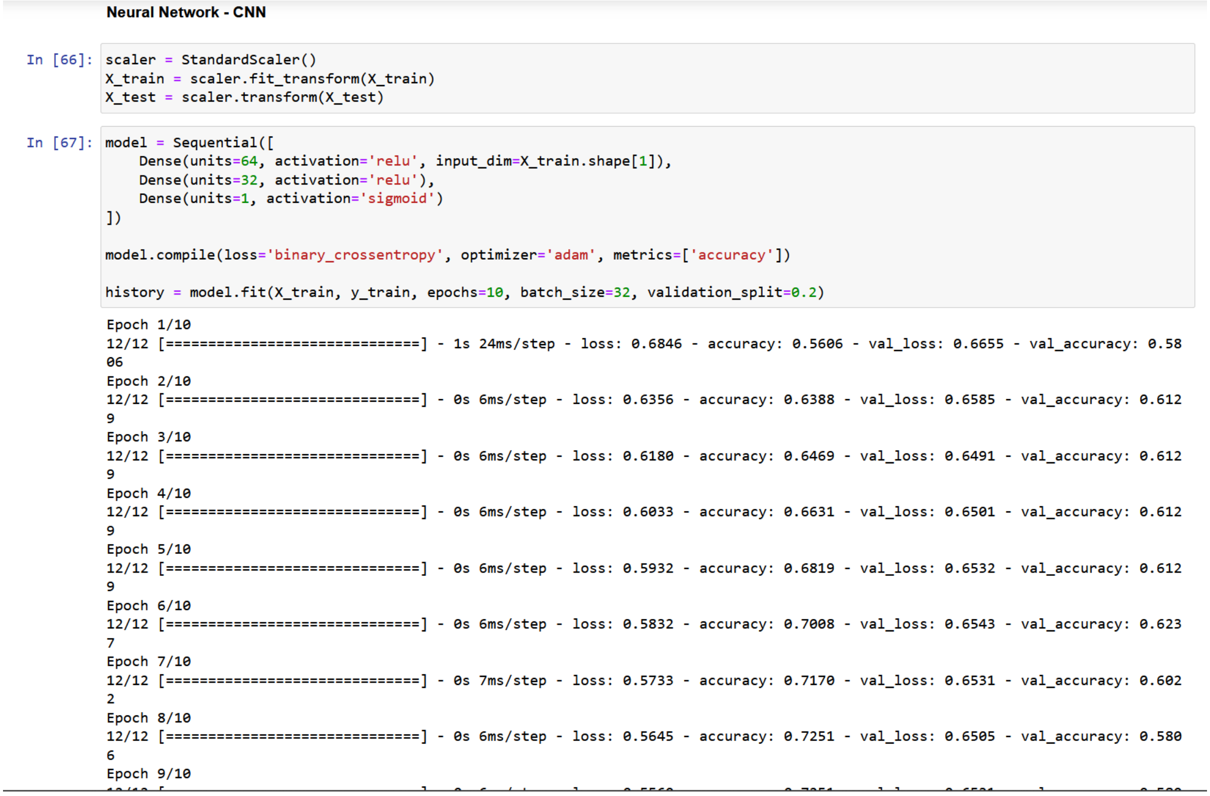


Support Vector Machine - SVM

```
In [65]: from sklearn.svm import SVC
         from sklearn.metrics import accuracy_score, classification_report

         svm_classifier = SVC(kernel='linear')

         svm_classifier.fit(X_train, y_train)

         svm_predictions = svm_classifier.predict(X_test)

         svm_accuracy = accuracy_score(y_test, svm_predictions)
         svm_report = classification_report(y_test, svm_predictions)

         print("Support Vector Machine (SVM) Classifier:")
         print(f"Accuracy: {svm_accuracy:.2f}")
         print("Classification Report:")
         print(svm_report)
```

```
Support Vector Machine (SVM) Classifier:
Accuracy: 0.65
Classification Report:
              precision    recall  f1-score   support

           0       0.65      1.00      0.79        76
           1       0.00      0.00      0.00        41

    accuracy                           0.65       117
   macro avg       0.32      0.50      0.39       117
weighted avg       0.42      0.65      0.51       117
```

Figure 27: Support Vector Machine (SVM) with scores



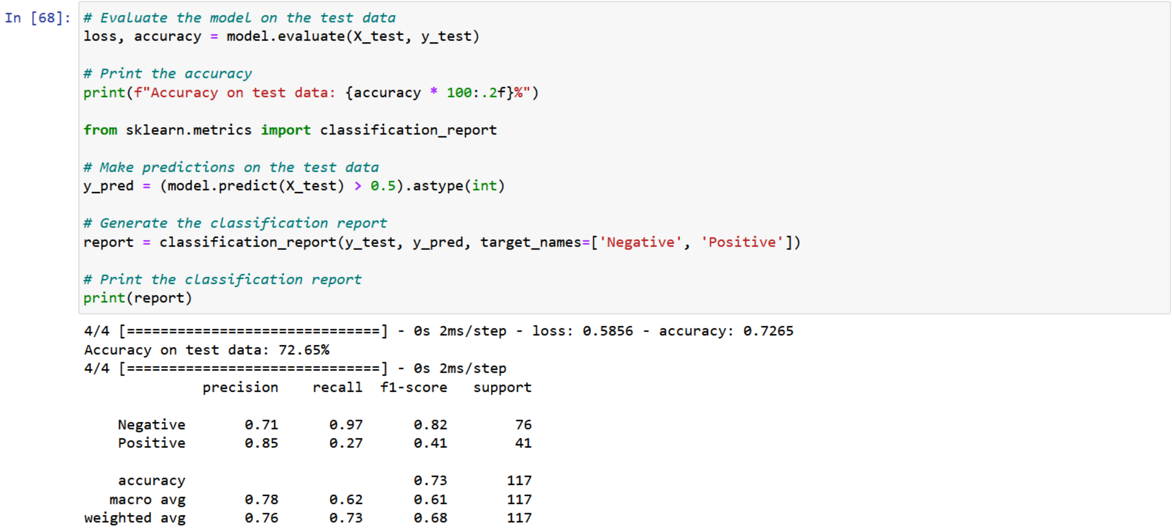
Neural Network - CNN

```
In [66]: scaler = StandardScaler()
         X_train = scaler.fit_transform(X_train)
         X_test = scaler.transform(X_test)
```

```
In [67]: model = Sequential([
             Dense(units=64, activation='relu', input_dim=X_train.shape[1]),
             Dense(units=32, activation='relu'),
             Dense(units=1, activation='sigmoid')
         ])

         model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

         history = model.fit(X_train, y_train, epochs=10, batch_size=32, validation_split=0.2)
```

```
Epoch 1/10
12/12 [==============================] - 1s 24ms/step - loss: 0.6846 - accuracy: 0.5606 - val_loss: 0.6655 - val_accuracy: 0.58
06
Epoch 2/10
12/12 [==============================] - 0s 6ms/step - loss: 0.6356 - accuracy: 0.6388 - val_loss: 0.6585 - val_accuracy: 0.612
9
Epoch 3/10
12/12 [==============================] - 0s 6ms/step - loss: 0.6180 - accuracy: 0.6469 - val_loss: 0.6491 - val_accuracy: 0.612
9
Epoch 4/10
12/12 [==============================] - 0s 6ms/step - loss: 0.6033 - accuracy: 0.6631 - val_loss: 0.6501 - val_accuracy: 0.612
9
Epoch 5/10
12/12 [==============================] - 0s 6ms/step - loss: 0.5932 - accuracy: 0.6819 - val_loss: 0.6532 - val_accuracy: 0.612
9
Epoch 6/10
12/12 [==============================] - 0s 6ms/step - loss: 0.5832 - accuracy: 0.7008 - val_loss: 0.6543 - val_accuracy: 0.623
7
Epoch 7/10
12/12 [==============================] - 0s 7ms/step - loss: 0.5733 - accuracy: 0.7170 - val_loss: 0.6531 - val_accuracy: 0.602
2
Epoch 8/10
12/12 [==============================] - 0s 6ms/step - loss: 0.5645 - accuracy: 0.7251 - val_loss: 0.6505 - val_accuracy: 0.580
6
Epoch 9/10
```

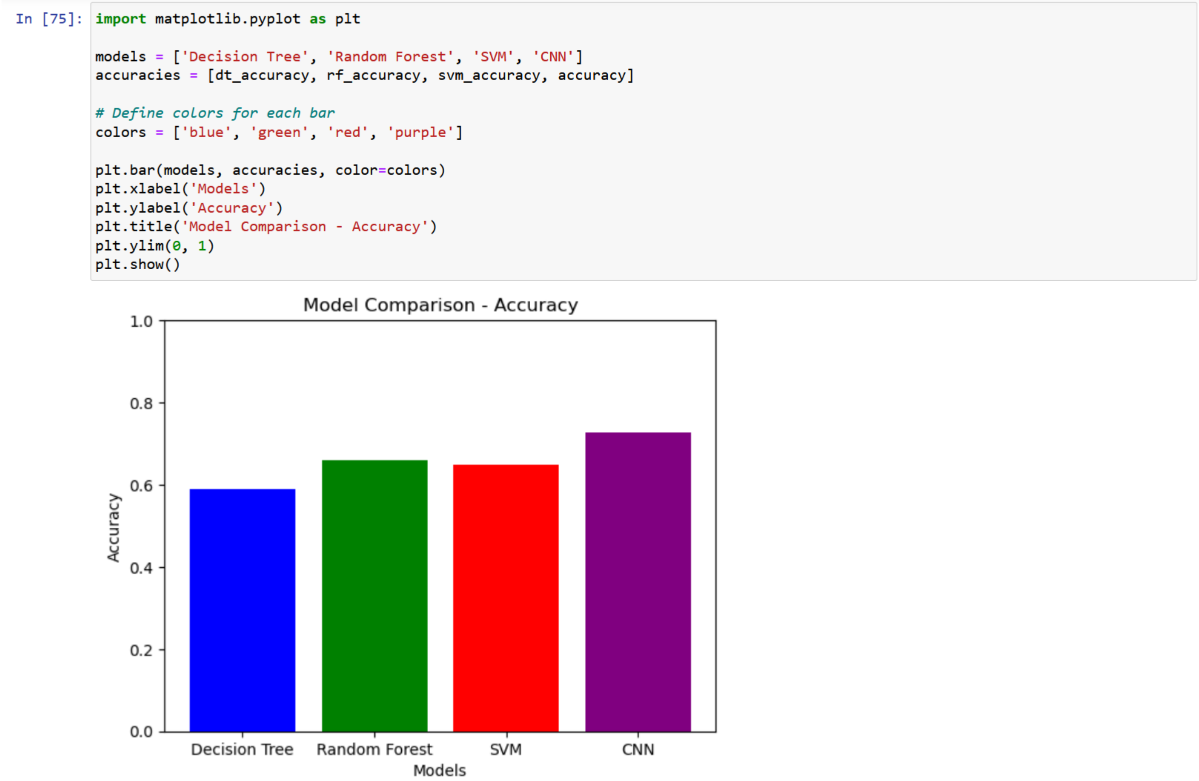
Figure 28: CNN and Epochs



```
In [68]: # Evaluate the model on the test data
         loss, accuracy = model.evaluate(X_test, y_test)

         # Print the accuracy
         print(f"Accuracy on test data: {accuracy * 100:.2f}%")

         from sklearn.metrics import classification_report

         # Make predictions on the test data
         y_pred = (model.predict(X_test) > 0.5).astype(int)

         # Generate the classification report
         report = classification_report(y_test, y_pred, target_names=['Negative', 'Positive'])

         # Print the classification report
         print(report)
```

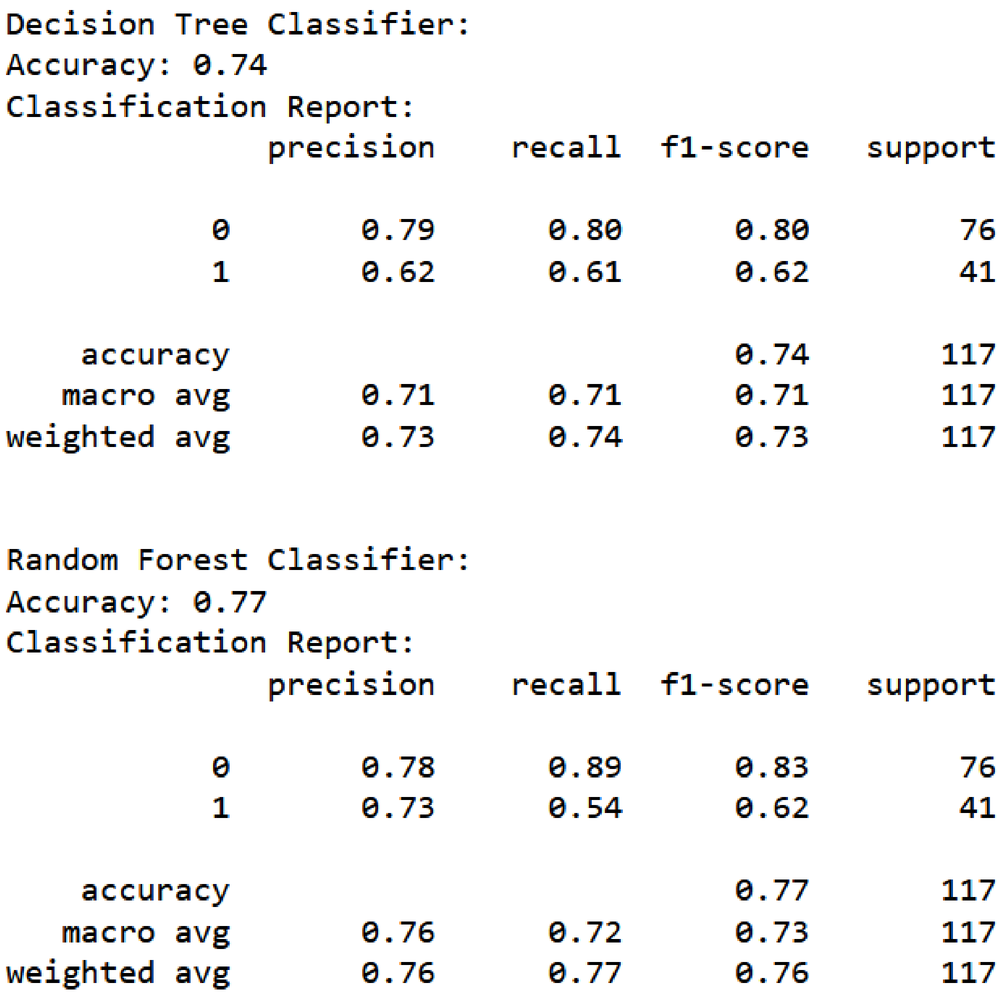
```
4/4 [==============================] - 0s 2ms/step - loss: 0.5856 - accuracy: 0.7265
Accuracy on test data: 72.65%
4/4 [==============================] - 0s 2ms/step
              precision    recall  f1-score   support

    Negative       0.71      0.97      0.82        76
    Positive       0.85      0.27      0.41        41

    accuracy                           0.73       117
   macro avg       0.78      0.62      0.61       117
weighted avg       0.76      0.73      0.68       117
```

Figure 29: CNN scores



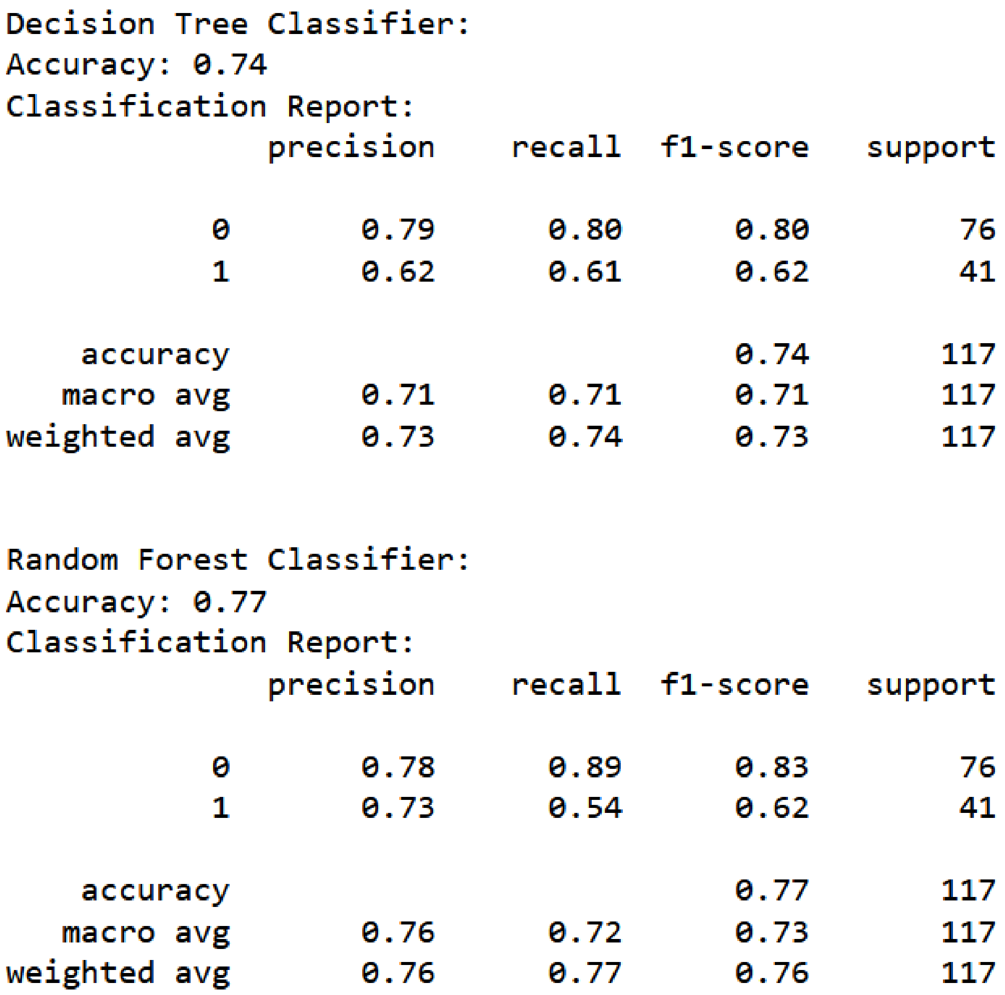
```
In [75]: import matplotlib.pyplot as plt

         models = ['Decision Tree', 'Random Forest', 'SVM', 'CNN']
         accuracies = [dt_accuracy, rf_accuracy, svm_accuracy, accuracy]

         # Define colors for each bar
         colors = ['blue', 'green', 'red', 'purple']

         plt.bar(models, accuracies, color=colors)
         plt.xlabel('Models')
         plt.ylabel('Accuracy')
         plt.title('Model Comparison - Accuracy')
         plt.ylim(0, 1)
         plt.show()
```

Figure 30: Case study 2 - Accuracy scores



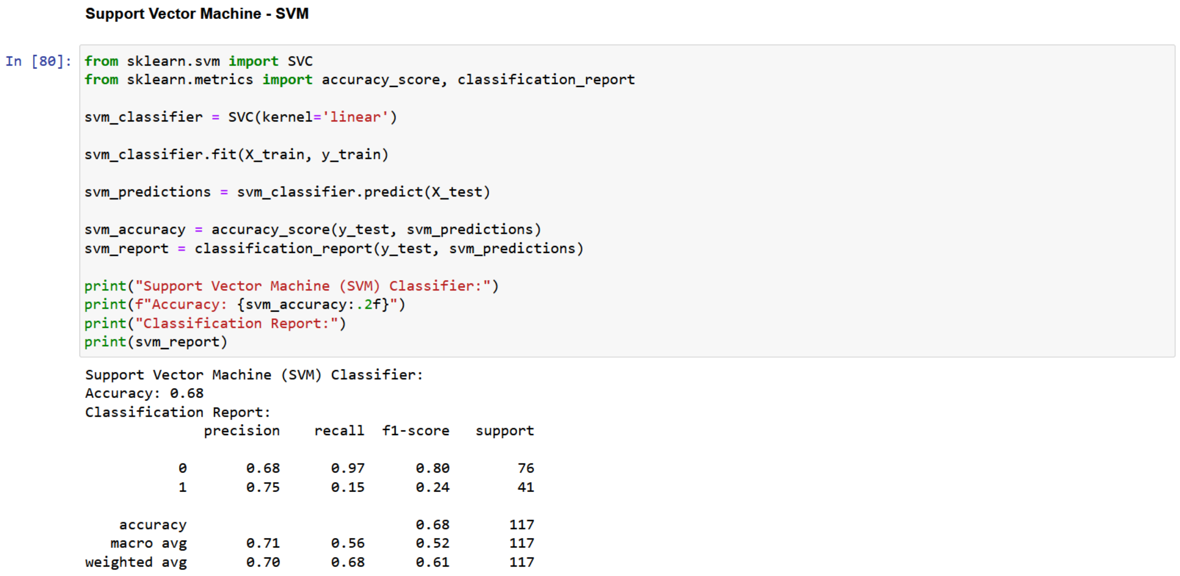
```
Decision Tree Classifier:
Accuracy: 0.74
Classification Report:
              precision    recall  f1-score   support

           0       0.79      0.80      0.80        76
           1       0.62      0.61      0.62        41

    accuracy                           0.74       117
   macro avg       0.71      0.71      0.71       117
weighted avg       0.73      0.74      0.73       117


Random Forest Classifier:
Accuracy: 0.77
Classification Report:
              precision    recall  f1-score   support

           0       0.78      0.89      0.83        76
           1       0.73      0.54      0.62        41

    accuracy                           0.77       117
   macro avg       0.76      0.72      0.73       117
weighted avg       0.76      0.77      0.76       117
```

Figure 31: Case Study 3



```
Decision Tree Classifier:
Accuracy: 0.74
Classification Report:
              precision    recall  f1-score   support

           0       0.79      0.80      0.80        76
           1       0.62      0.61      0.62        41

    accuracy                           0.74       117
   macro avg       0.71      0.71      0.71       117
weighted avg       0.73      0.74      0.73       117


Random Forest Classifier:
Accuracy: 0.77
Classification Report:
              precision    recall  f1-score   support

           0       0.78      0.89      0.83        76
           1       0.73      0.54      0.62        41

    accuracy                           0.77       117
   macro avg       0.76      0.72      0.73       117
weighted avg       0.76      0.77      0.76       117
```

Figure 32: Scores of Random Forest and Decision Tree

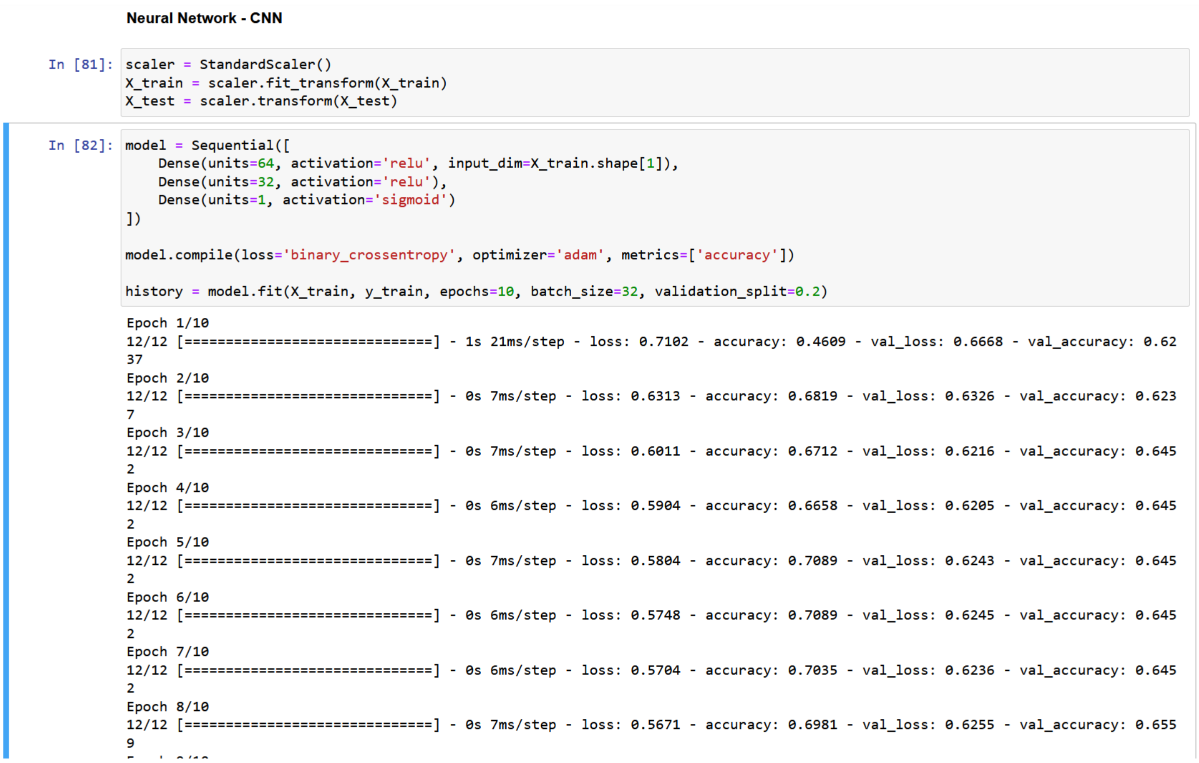


Support Vector Machine - SVM

```
In [80]: from sklearn.svm import SVC
         from sklearn.metrics import accuracy_score, classification_report

         svm_classifier = SVC(kernel='linear')

         svm_classifier.fit(X_train, y_train)

         svm_predictions = svm_classifier.predict(X_test)

         svm_accuracy = accuracy_score(y_test, svm_predictions)
         svm_report = classification_report(y_test, svm_predictions)

         print("Support Vector Machine (SVM) Classifier:")
         print(f"Accuracy: {svm_accuracy:.2f}")
         print("Classification Report:")
         print(svm_report)
```

```
Support Vector Machine (SVM) Classifier:
Accuracy: 0.68
Classification Report:
              precision    recall  f1-score   support

           0       0.68      0.97      0.80        76
           1       0.75      0.15      0.24        41

    accuracy                           0.68       117
   macro avg       0.71      0.56      0.52       117
weighted avg       0.70      0.68      0.61       117
```

Figure 33: SVM with scores



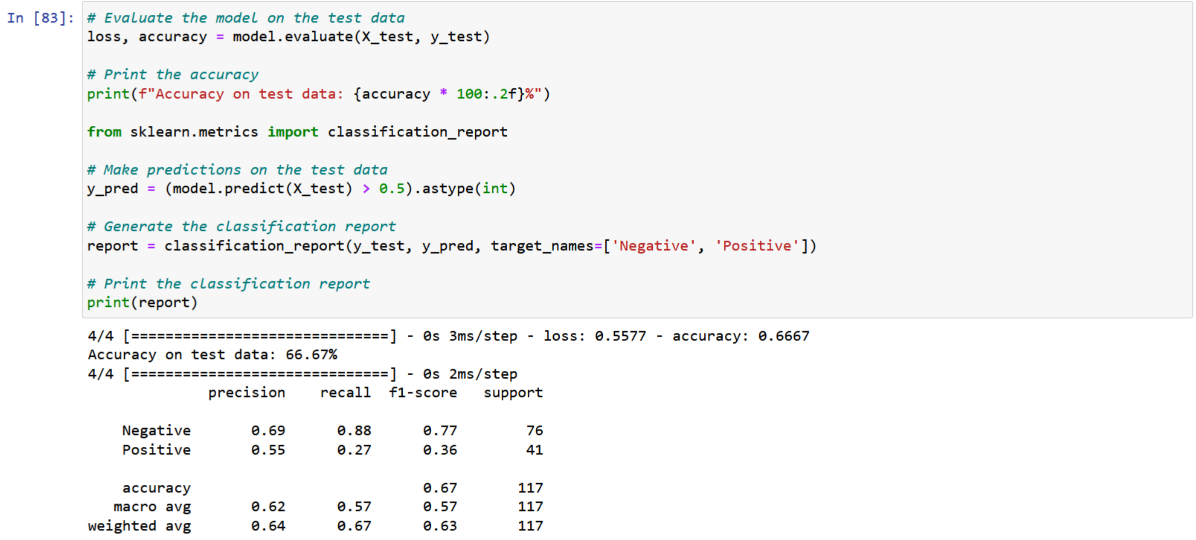
Neural Network - CNN

```
In [81]: scaler = StandardScaler()
         X_train = scaler.fit_transform(X_train)
         X_test = scaler.transform(X_test)
```

```
In [82]: model = Sequential([
             Dense(units=64, activation='relu', input_dim=X_train.shape[1]),
             Dense(units=32, activation='relu'),
             Dense(units=1, activation='sigmoid')
         ])

         model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

         history = model.fit(X_train, y_train, epochs=10, batch_size=32, validation_split=0.2)
```

```
Epoch 1/10
12/12 [==============================] - 1s 21ms/step - loss: 0.7102 - accuracy: 0.4609 - val_loss: 0.6668 - val_accuracy: 0.62
37
Epoch 2/10
12/12 [==============================] - 0s 7ms/step - loss: 0.6313 - accuracy: 0.6819 - val_loss: 0.6326 - val_accuracy: 0.623
7
Epoch 3/10
12/12 [==============================] - 0s 7ms/step - loss: 0.6011 - accuracy: 0.6712 - val_loss: 0.6216 - val_accuracy: 0.645
2
Epoch 4/10
12/12 [==============================] - 0s 6ms/step - loss: 0.5904 - accuracy: 0.6658 - val_loss: 0.6205 - val_accuracy: 0.645
2
Epoch 5/10
12/12 [==============================] - 0s 7ms/step - loss: 0.5804 - accuracy: 0.7089 - val_loss: 0.6243 - val_accuracy: 0.645
2
Epoch 6/10
12/12 [==============================] - 0s 6ms/step - loss: 0.5748 - accuracy: 0.7089 - val_loss: 0.6245 - val_accuracy: 0.645
2
Epoch 7/10
12/12 [==============================] - 0s 6ms/step - loss: 0.5704 - accuracy: 0.7035 - val_loss: 0.6236 - val_accuracy: 0.645
2
Epoch 8/10
12/12 [==============================] - 0s 7ms/step - loss: 0.5671 - accuracy: 0.6981 - val_loss: 0.6255 - val_accuracy: 0.655
9
```

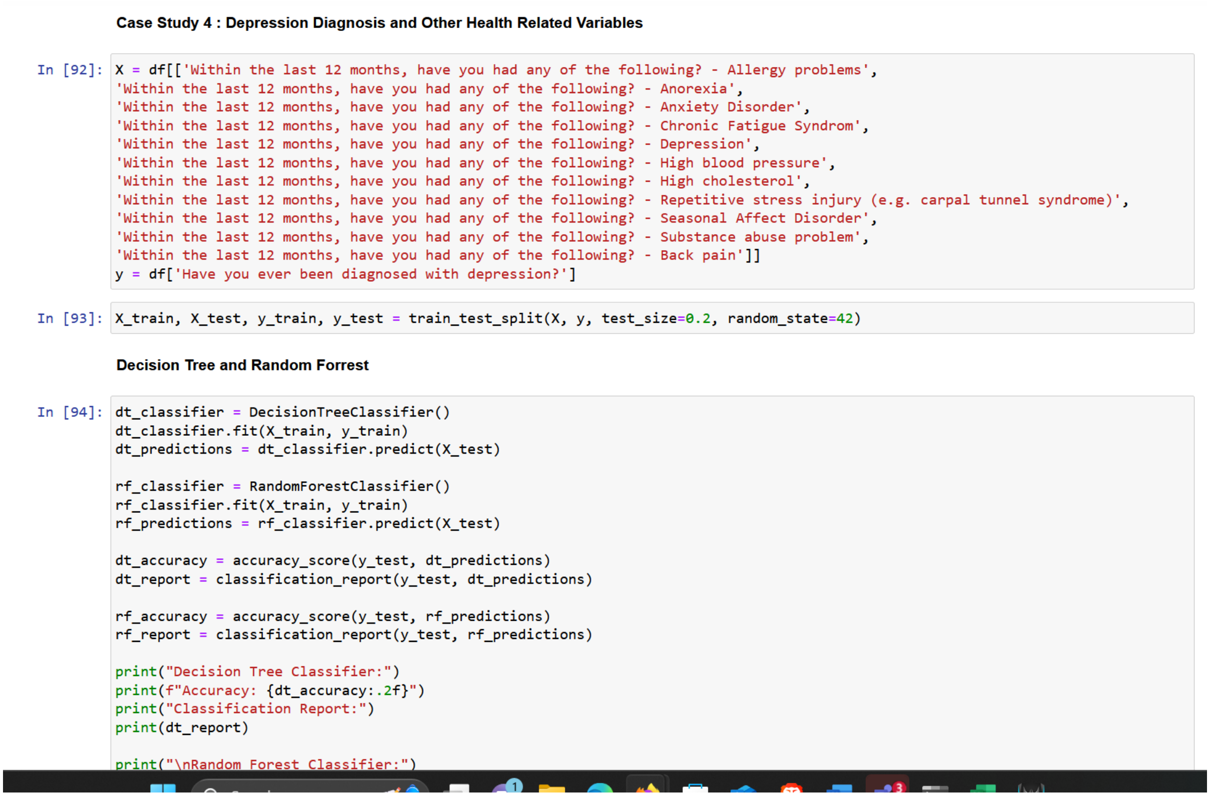
Figure 34: CNN with Epochs



```
In [83]: # Evaluate the model on the test data
         loss, accuracy = model.evaluate(X_test, y_test)

         # Print the accuracy
         print(f"Accuracy on test data: {accuracy * 100:.2f}%")

         from sklearn.metrics import classification_report

         # Make predictions on the test data
         y_pred = (model.predict(X_test) > 0.5).astype(int)

         # Generate the classification report
         report = classification_report(y_test, y_pred, target_names=['Negative', 'Positive'])

         # Print the classification report
         print(report)
```

```
4/4 [==============================] - 0s 3ms/step - loss: 0.5577 - accuracy: 0.6667
Accuracy on test data: 66.67%
4/4 [==============================] - 0s 2ms/step
              precision    recall  f1-score   support

    Negative       0.69      0.88      0.77        76
    Positive       0.55      0.27      0.36        41

    accuracy                           0.67       117
   macro avg       0.62      0.57      0.57       117
weighted avg       0.64      0.67      0.63       117
```

Figure 35: CNN scores



Case Study 4 : Depression Diagnosis and Other Health Related Variables

In [92]:

```
X = df[['Within the last 12 months, have you had any of the following? - Allergy problems',
'Within the last 12 months, have you had any of the following? - Anorexia',
'Within the last 12 months, have you had any of the following? - Anxiety Disorder',
'Within the last 12 months, have you had any of the following? - Chronic Fatigue Syndrom',
'Within the last 12 months, have you had any of the following? - Depression',
'Within the last 12 months, have you had any of the following? - High blood pressure',
'Within the last 12 months, have you had any of the following? - High cholesterol',
'Within the last 12 months, have you had any of the following? - Repetitive stress injury (e.g. carpal tunnel syndrome)',
'Within the last 12 months, have you had any of the following? - Seasonal Affect Disorder',
'Within the last 12 months, have you had any of the following? - Substance abuse problem',
'Within the last 12 months, have you had any of the following? - Back pain']]
y = df['Have you ever been diagnosed with depression?']
```

In [93]:

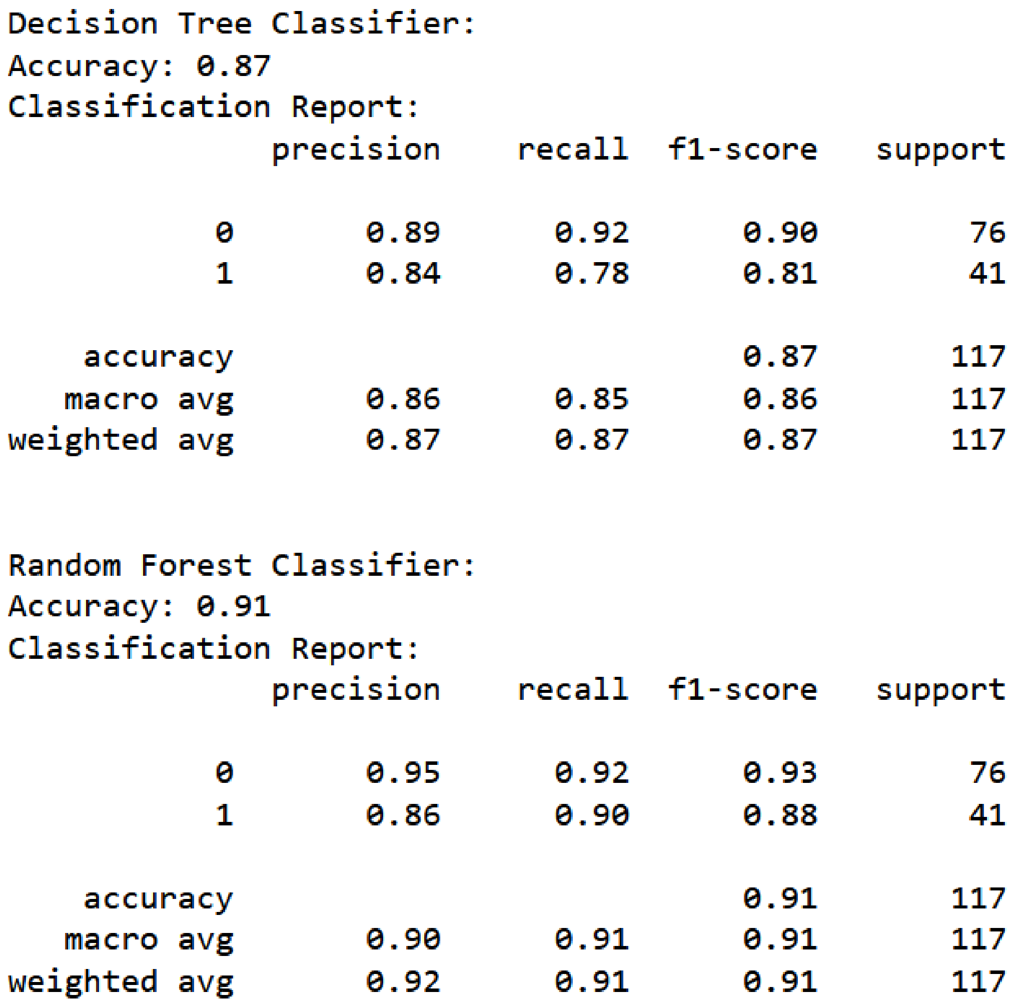
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Decision Tree and Random Forrest

In [94]:

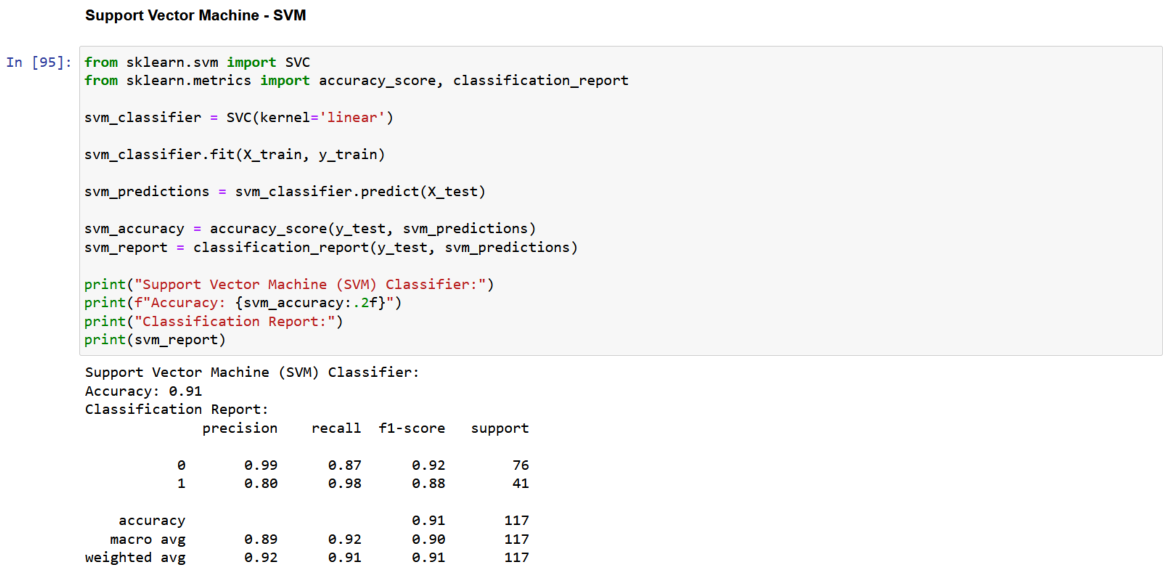
```
dt_classifier = DecisionTreeClassifier()
dt_classifier.fit(X_train, y_train)
dt_predictions = dt_classifier.predict(X_test)

rf_classifier = RandomForestClassifier()
rf_classifier.fit(X_train, y_train)
rf_predictions = rf_classifier.predict(X_test)

dt_accuracy = accuracy_score(y_test, dt_predictions)
dt_report = classification_report(y_test, dt_predictions)

rf_accuracy = accuracy_score(y_test, rf_predictions)
rf_report = classification_report(y_test, rf_predictions)

print("Decision Tree Classifier:")
print(f"Accuracy: {dt_accuracy:.2f}")
print("Classification Report:")
print(dt_report)

print("\nRandom Forest Classifier:")
```

Figure 36: Case Study 4



```
Decision Tree Classifier:
Accuracy: 0.87
Classification Report:
              precision    recall  f1-score   support

           0       0.89      0.92      0.90        76
           1       0.84      0.78      0.81        41

    accuracy                           0.87       117
   macro avg       0.86      0.85      0.86       117
weighted avg       0.87      0.87      0.87       117


Random Forest Classifier:
Accuracy: 0.91
Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.92      0.93        76
           1       0.86      0.90      0.88        41

    accuracy                           0.91       117
   macro avg       0.90      0.91      0.91       117
weighted avg       0.92      0.91      0.91       117
```

Figure 37: Decision Tree and Random forest scores

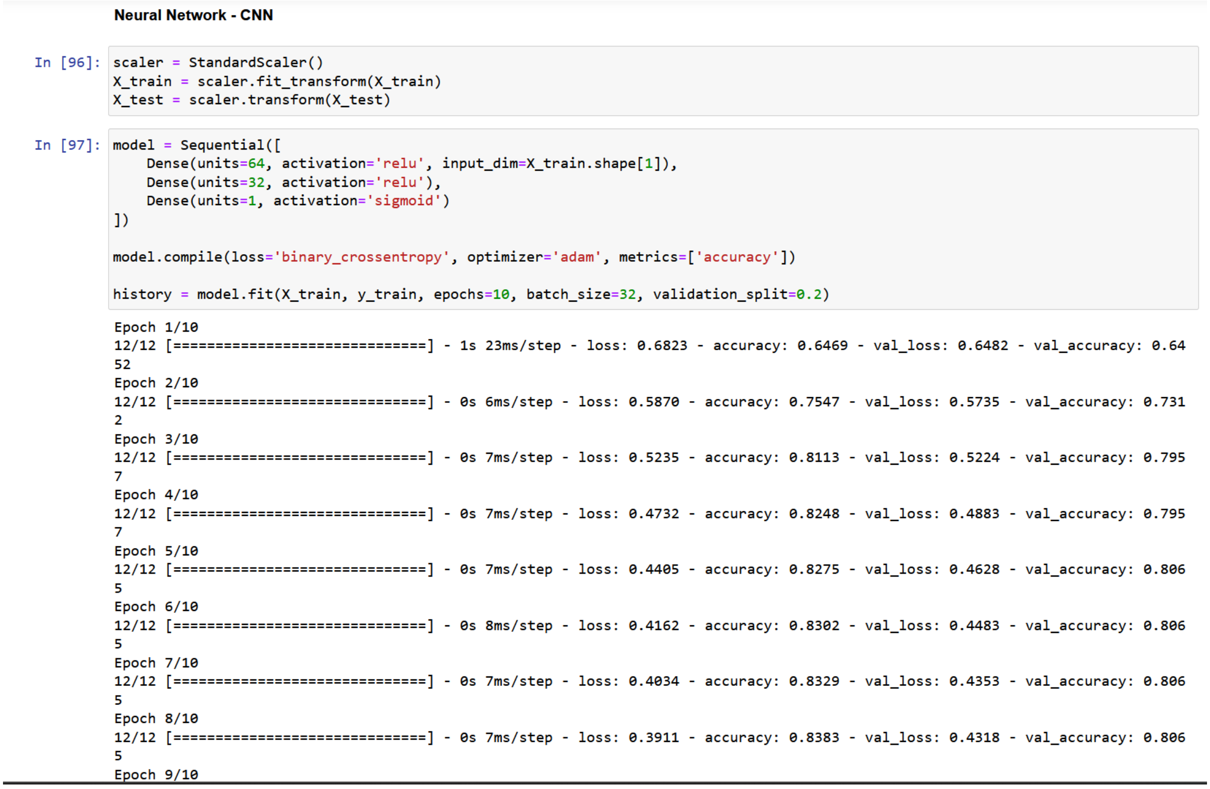


Support Vector Machine - SVM

```
In [95]: from sklearn.svm import SVC
         from sklearn.metrics import accuracy_score, classification_report

         svm_classifier = SVC(kernel='linear')

         svm_classifier.fit(X_train, y_train)

         svm_predictions = svm_classifier.predict(X_test)

         svm_accuracy = accuracy_score(y_test, svm_predictions)
         svm_report = classification_report(y_test, svm_predictions)

         print("Support Vector Machine (SVM) Classifier:")
         print(f"Accuracy: {svm_accuracy:.2f}")
         print("Classification Report:")
         print(svm_report)
```

```
Support Vector Machine (SVM) Classifier:
Accuracy: 0.91
Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.87      0.92        76
           1       0.80      0.98      0.88        41

    accuracy                           0.91       117
   macro avg       0.89      0.92      0.90       117
weighted avg       0.92      0.91      0.91       117
```

Figure 38: SVM with scores



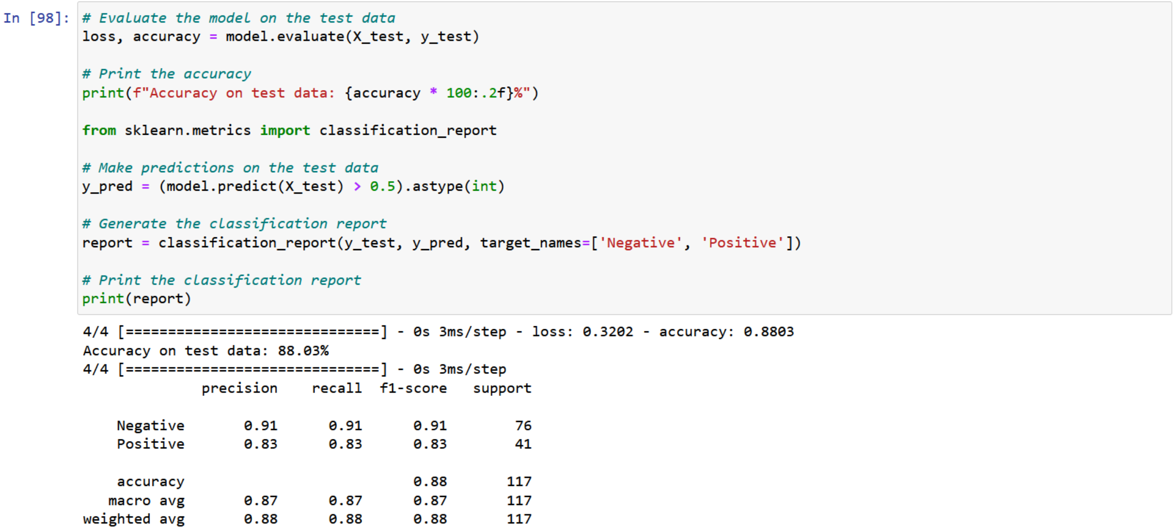
Neural Network - CNN

```
In [96]: scaler = StandardScaler()
         X_train = scaler.fit_transform(X_train)
         X_test = scaler.transform(X_test)
```

```
In [97]: model = Sequential([
             Dense(units=64, activation='relu', input_dim=X_train.shape[1]),
             Dense(units=32, activation='relu'),
             Dense(units=1, activation='sigmoid')
         ])

         model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

         history = model.fit(X_train, y_train, epochs=10, batch_size=32, validation_split=0.2)
```

```
Epoch 1/10
12/12 [==============================] - 1s 23ms/step - loss: 0.6823 - accuracy: 0.6469 - val_loss: 0.6482 - val_accuracy: 0.64
52
Epoch 2/10
12/12 [==============================] - 0s 6ms/step - loss: 0.5870 - accuracy: 0.7547 - val_loss: 0.5735 - val_accuracy: 0.731
2
Epoch 3/10
12/12 [==============================] - 0s 7ms/step - loss: 0.5235 - accuracy: 0.8113 - val_loss: 0.5224 - val_accuracy: 0.795
7
Epoch 4/10
12/12 [==============================] - 0s 7ms/step - loss: 0.4732 - accuracy: 0.8248 - val_loss: 0.4883 - val_accuracy: 0.795
7
Epoch 5/10
12/12 [==============================] - 0s 7ms/step - loss: 0.4405 - accuracy: 0.8275 - val_loss: 0.4628 - val_accuracy: 0.806
5
Epoch 6/10
12/12 [==============================] - 0s 8ms/step - loss: 0.4162 - accuracy: 0.8302 - val_loss: 0.4483 - val_accuracy: 0.806
5
Epoch 7/10
12/12 [==============================] - 0s 7ms/step - loss: 0.4034 - accuracy: 0.8329 - val_loss: 0.4353 - val_accuracy: 0.806
5
Epoch 8/10
12/12 [==============================] - 0s 7ms/step - loss: 0.3911 - accuracy: 0.8383 - val_loss: 0.4318 - val_accuracy: 0.806
5
Epoch 9/10
```

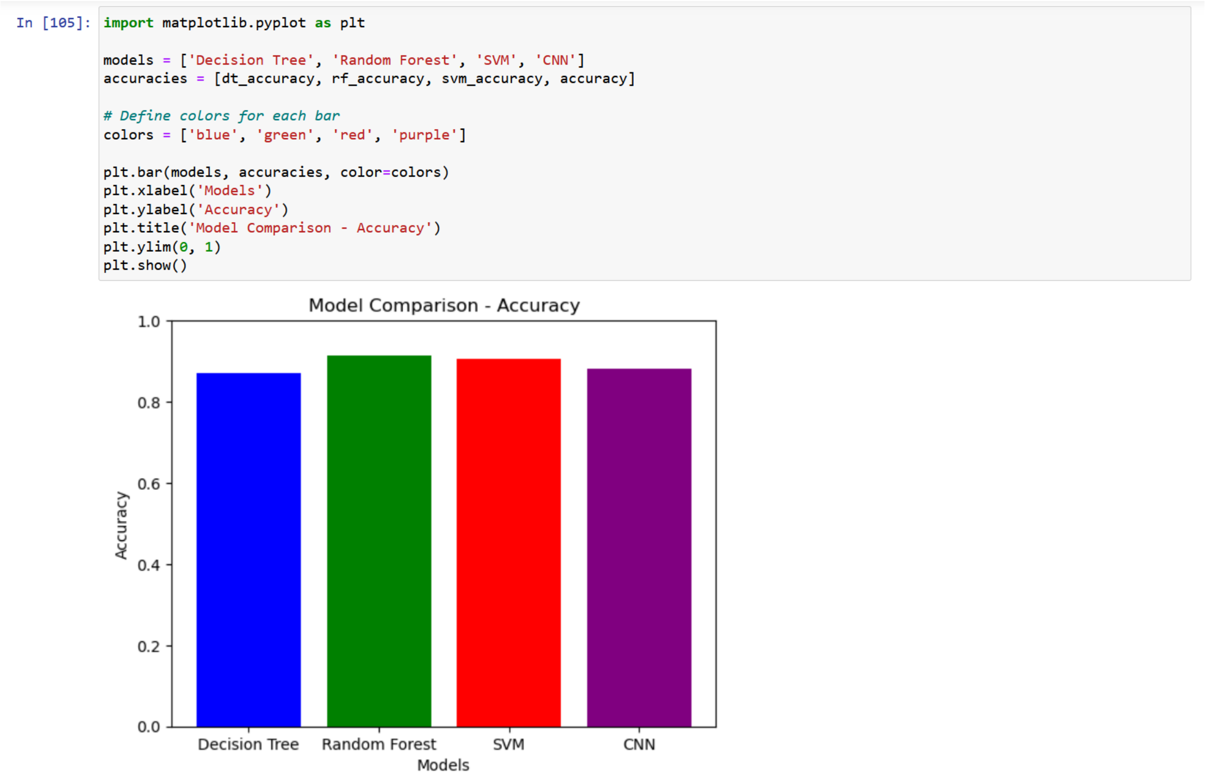
Figure 39: CNN with Epochs



```
In [98]: # Evaluate the model on the test data
         loss, accuracy = model.evaluate(X_test, y_test)

         # Print the accuracy
         print(f"Accuracy on test data: {accuracy * 100:.2f}%")

         from sklearn.metrics import classification_report

         # Make predictions on the test data
         y_pred = (model.predict(X_test) > 0.5).astype(int)

         # Generate the classification report
         report = classification_report(y_test, y_pred, target_names=['Negative', 'Positive'])

         # Print the classification report
         print(report)
```

```
4/4 [==============================] - 0s 3ms/step - loss: 0.3202 - accuracy: 0.8803
Accuracy on test data: 88.03%
4/4 [==============================] - 0s 3ms/step
              precision    recall  f1-score   support

    Negative       0.91      0.91      0.91        76
    Positive       0.83      0.83      0.83        41

    accuracy                           0.88       117
   macro avg       0.87      0.87      0.87       117
weighted avg       0.88      0.88      0.88       117
```

Figure 40: CNN scores



```
In [105]: import matplotlib.pyplot as plt

          models = ['Decision Tree', 'Random Forest', 'SVM', 'CNN']
          accuracies = [dt_accuracy, rf_accuracy, svm_accuracy, accuracy]

          # Define colors for each bar
          colors = ['blue', 'green', 'red', 'purple']

          plt.bar(models, accuracies, color=colors)
          plt.xlabel('Models')
          plt.ylabel('Accuracy')
          plt.title('Model Comparison - Accuracy')
          plt.ylim(0, 1)
          plt.show()
```

Figure 41: Case Study 4 - Accuracy comparison