Configuration Manual

MSc Research Project
Data Analytics

Shruthi Manthena
Student ID: x23235853

School of Computing
National College of Ireland

Supervisor: Teerath Kumar Menghwar

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Shruthi Manthena		
Student ID:	x23235853		
Programme:	MSc Data Analytics	**Year:**	2024 - 25
Module:	MSc Research Project		
Lecturer:	Teerath Kumar Menghwar		
Submission Due Date:	12/12/2024		
Project Title:	A Comparative Study of Machine Learning Algorithms for Vehicle Insurance Fraud Detection		

Word Count: 979 **Page Count:** 9

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: Shruthi Manthena

Date: 11/12/2024

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

Shruthi Manthena
Student ID: x23235853

1. Introduction

This configuration manual is a step-by-step guide of the experiment setup and result of the machine learning study involving vehicle insurance fraud detection. The research assesses datasets and machine learning algorithms for prediction of claims and identification of fraud. This research evaluates various Machine learning models Decision Tree, K- Nearest Neighbors (KNN), Light Gradient Boosting Machine (Light GBM), Random Forest and Support Vector Classifier (SVC). This manual comprises all the technical details include Software, Hardware infrastructure, Python libraries used and related packages for visualizations as well. It also includes Equipment specifications, data pre-processing, algorithms configurations, besides performance assessment. This guide provides step-by-step instructions which would help and understand the users to configure the environment, to replicate the experimental setup, do the case experiment and to analyse the results. This document gives details of the software, packages, and configurations required to ensure that it gives similar experimental environment therefore similar results.

2. Development Environment

For this Research study, the environment used is Mac OS. Specifications of both Hardware and Software are explained in detail below. The datasets used for the project are – Insurance claims and Car Insurance claims.

2.1 Hardware Specifications

- OS: Mac OS
- Chip: Apple M1

The environment was created using the local hardware system specifications mentioned above. Furthermore, it is not necessary to have the identical specifications to recreate the setting in order to conduct the experiment or rerun the setup.

2.2 Software Specifications

- Operating System: Mac OS or any other operating system can be used.
  Ex: Windows 10/11 or Ubuntu 20.04+
- Programming Language: Python version 3.11.7

```
[2]: !python --version
Python 3.11.7
```

Figure 1: Python Version

- Integrated Development Environment (IDE): Jupyter Notebook 6.5.4 or higher version.

2.3 Python Libraries Required

Figure 2 shows the list of necessary Python Libraries required for execution of the code and these python libraries can be installed using pip command.

- Numpy
- Pandas
- Scikit-learn
- Matplotlib
- Seaborn
- Keras

```
# importing the libraries
import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from lightgbm import LGBMClassifier
from sklearn.metrics import accuracy_score, f1_score, precision_score, recall_score, confusion_matrix, classification_report
```

Figure 2: Libraries Imported (common for all models)

3. Data Source

Total three datasets are used for this project and are obtained from Kaggle and include insurance fraud claims with key attributes such as age, policy number, policy state, months as customer and more.

- **Claim Fraud Identification:** Includes insurance fraud labels and associated features. https://www.kaggle.com/code/buntyshah/insurance-fraud-claims-detection/input?select=insurance_claims.csv

months_as_customer	age	policy_number	policy_bind_date	policy_state	policy_csl	policy_deductable	policy_annual_premium	umbrella_limit	insured_zip	insured_sex	insured_education_level	insured_occupation
328	48	521585	2014-10-17	OH	250/500	1000	1406.91	0	466132	MALE	MD	craft-repair
228	42	342868	2006-06-27	IN	250/500	2000	1197.22	5000000	468176	MALE	MD	machine-op-inspct
134	29	687698	2000-09-06	OH	100/300	2000	1413.14	5000000	430632	FEMALE	PhD	sales
256	41	227811	1990-05-25	IL	250/500	2000	1415.74	6000000	608117	FEMALE	PhD	armed-forces
228	44	367455	2014-06-06	IL	500/1000	1000	1583.91	6000000	610706	MALE	Associate	sales
256	39	104594	2006-10-12	OH	250/500	1000	1351.1	0	478456	FEMALE	PhD	tech-support
137	34	413978	2000-06-04	IN	250/500	1000	1333.35	0	441716	MALE	PhD	prof-specialty

Figure 3: Dataset 1 columns and details

- **Automobile Insurance Data:** Records insurance policy and claim details. https://www.kaggle.com/datasets/sagnik1511/car-insurance-data

ID	AGE	GENDER	RACE	DRIVING_EXPERIENCE	EDUCATION	INCOME	CREDIT_SCORE	VEHICLE_OWNERSHIP	VEHICLE_YEAR	MARRIED	CHILDREN	POSTAL_CODE	ANNUAL_MILEAGE	VEHICLE_T
569520	65	female	majority	0-9y	high school	upper class	0.629027313918201	1.0	after 2015	0.0	1.0	10238	12000.0	sedan
750365	16-25	male	majority	0-9y	none	poverty	0.3577571170184630	0.0	before 2015	0.0	0.0	10238	16000.0	sedan
199901	16-25	female	majority	0-9y	high school	working class	0.4931457852181980	1.0	before 2015	0.0	0.0	10238	11000.0	sedan
478866	16-25	male	majority	0-9y	university	working class	0.2060128507324560	1.0	before 2015	0.0	1.0	32765	11000.0	sedan
731664	26-39	male	majority	10-19y	none	working class	0.3883658881572180	1.0	before 2015	0.0	0.0	32765	12000.0	sedan
877557	40-64	female	majority	20-29y	high school	upper class	0.6191273725847390	1.0	after 2015	0.0	1.0	10238	13000.0	sedan
930134	65	male	majority	30y+	high school	upper class	0.4929435502195340	0.0	after 2015	1.0	1.0	10238	13000.0	sedan

Figure 4: Dataset 2 columns and details

- **Vehicle Insurance Claim Prediction:** Contains historical claim records for predictive modeling. https://www.kaggle.com/code/has9800/vehicle-insurance-claim-prediction-98-99/input

ID	KIDSDRIV	BIRTH	AGE	HOMEKIDS	YOJ	INCOME	PARENT1	HOME_VAL	MSTATUS	GENDER	EDUCATION	OCCUPATION	TRAVTIME	CAR_USE	BLUEBOOK	TIF	CAR_TYPE	RED_CAR
63581743	0	16MAR39	60	0	11	$ 67,349.00	No	$ 0.00	z_No	M	PhD	Professional	14	Private	$ 14,230.00	11	Minivan	yes
132761049	0	21JAN56	43	0	11	$ 91,449.00	No	$ 2,57,252.00	z_No	M	z_High School	z_Blue Collar	22	Commercial	$ 14,940.00	1	Minivan	yes
921317019	0	18NOV51	48	0	11	$ 52,881.00	No	$ 0.00	z_No	M	Bachelors	Manager	26	Private	$ 21,970.00	1	Van	yes
727598473	0	05MAR64	35	1	10	$ 16,039.00	No	$ 1,24,191.00	Yes	z_F	z_High School	Clerical	5	Private	$ 4,010.00	4	z_SUV	no
450221861	0	05JUN48	51	0	14		No	$ 3,06,251.00	Yes	M	<High School	z_Blue Collar	32	Private	$ 15,440.00	7	Minivan	yes
743146596	0	17MAY49	50	0		$ 1,14,986.00	No	$ 2,43,925.00	Yes	z_F	PhD	Doctor	36	Private	$ 18,000.00	1	z_SUV	no
871024631	0	05MAY65	34	1	12	$ 1,25,301.00	Yes	$ 0.00	z_No	z_F	Bachelors	z_Blue Collar	46	Commercial	$ 17,430.00	1	Sports Car	no
792300541	0	28FEB45	54	0		$ 18,755.00	No		Yes	z_F	<High School	z_Blue Collar	33	Private	$ 8,780.00	1	z_SUV	no

Figure 5: Dataset 3 columns and details

4. Project Code Files

- **Data Cleaning and Preprocessing:** Handles missing data, encodes categorical features and scales numerical values.
- **Model Implementation:** Includes code for training Decision Tree, KNN, Light GBM, Random Forest and SVC models.
- **Performance Evaluation:** Calculates accuracy, F1 score, precision and recall for each model and dataset.
- **Results Comparison:** Compares the efficiency of models across datasets.

EDA_Dataset1.ipynb
EDA_Dataset2.ipynb
EDA_Dataset3.ipynb
model-training-dataset1.ipynb
model-training-dataset2.ipynb
model-training-dataset3.ipynb

Figure 6: Code files of EDA and model training of all 3 datasets

5. Data Preparation

5.1 Extracting Data

Loading the datasets from CSV file uploaded:

```
import pandas as pd
dataset1 = pd.read_csv('Dataset1.csv')
info1 = dataset1.info()
head1 = dataset1.head()
(info1, head1)
```

```
import pandas as pd

# Load the datasets
dataset2 = pd.read_csv('Dataset2.csv')

# Display basic info and first few rows of each dataset
info2 = dataset2.info()
head2 = dataset2.head()

(info2, head2)
```

```
import pandas as pd
dataset3 = pd.read_csv('Dataset3.csv')
dataset3.info()
dataset3.head()
```

Figure 7: Datasets Imported

5.2 Data Pre-processing

- Handling Missing Values: Impute missing data using mean/median or interpolate.
- Data Separation: separating the data variables.
- This format is repeated for every model building code as well as EDA.

```
# Dropping unnecessary columns
data = df_resampled.drop(['PLANT_ID', 'SOURCE_KEY_gen', 'SOURCE_KEY_weather', 'hour'], axis=1)

# Splitting the data into train and test sets
train_data, test_data = train_test_split(data, test_size=0.2, random_state=42)

# Separating the target variable
X_train = train_data.drop('TOTAL_YIELD', axis=1)
y_train = train_data['TOTAL_YIELD']
X_test = test_data.drop('TOTAL_YIELD', axis=1)
y_test = test_data['TOTAL_YIELD']

# Building a RandomForest model
rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)

# Predicting on the test set
y_pred = rf_model.predict(X_test)

# Evaluating the model
mse = mean_squared_error(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

mse, mae, r2
```

Figure 8: Handling unnecessary columns

```
# filling null values in credit score and annual age column with mean impute
df2["CREDIT_SCORE"] = df2["CREDIT_SCORE"].fillna(df2["CREDIT_SCORE"].mean())
df2["ANNUAL_MILEAGE"] = df2["ANNUAL_MILEAGE"].fillna(df2["ANNUAL_MILEAGE"].mean())

le = LabelEncoder()
le_count = 0
droplist = []

# Iterate through the columns
for col in df2:
    if df2[col].dtype == 'object':
        print(col, len(df2[col].unique()))
        # If 2 or fewer unique categories
        if len(list(df2[col].unique())) <= 4:
            # Train on the training data
            le.fit(df2[col])
            # Transform both training and testing data
            df2[col] = le.transform(df2[col])

            # Keep track of how many columns were label encoded
            le_count += 1
        else:
            droplist.append(col)

print('%d columns were label encoded.' % le_count)
df2.drop(columns=droplist, inplace=True)
```

Figure 9: Filling null values and encoding

5.3 Data Splitting

- Divide data into training (70%) and testing (30%) sets

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3) # train test split
```

Figure 10: Splitting training and testing data

6. Model Building

This section covers list of models used, training of the models and Evaluation of models performance using the metrics calculation of the same.

Models trained and used:

- Light GBM
- KNN
- Decision Tree
- Random Forest
- SVC

```python
# Initialize classifiers
classifiers = {
    'Random Forest': RandomForestClassifier(),
    'LightGBM': LGBMClassifier(),
    'KNN': KNeighborsClassifier(),
    'SVC': SVC(),
    'Decision Tree': DecisionTreeClassifier()
}

# Dictionary to store evaluation metrics
metrics = {}

# Training and evaluation of each classifier
for name, clf in classifiers.items():
    # Train the model
    clf.fit(X_train, y_train)
    # Predict on the test set
    y_pred = clf.predict(X_test)

    # Calculate metrics
    accuracy = accuracy_score(y_test, y_pred)
    f1 = f1_score(y_test, y_pred)
    precision = precision_score(y_test, y_pred)
    recall = recall_score(y_test, y_pred)
    confusion = confusion_matrix(y_test, y_pred)

    # Store metrics in the dictionary
    metrics[name] = {
        'Accuracy': accuracy,
        'F1 Score': f1,
        'Precision': precision,
        'Recall': recall,
        'Confusion Matrix': confusion,
    }
```

Figure 11: Model training code snippet

```python
import matplotlib.pyplot as plt
import numpy as np

# Prepare data for plotting
models = list(metrics.keys())
accuracy = [metrics[model]['Accuracy'] for model in models]
f1_score = [metrics[model]['F1 Score'] for model in models]
precision = [metrics[model]['Precision'] for model in models]
recall = [metrics[model]['Recall'] for model in models]

# Set bar width and positions
bar_width = 0.2
x = np.arange(len(models))

# Plot grouped bar chart
plt.figure(figsize=(12, 6))
plt.bar(x - 1.5 * bar_width, accuracy, width=bar_width, label='Accuracy', alpha=0.7)
plt.bar(x - 0.5 * bar_width, f1_score, width=bar_width, label='F1 Score', alpha=0.7)
plt.bar(x + 0.5 * bar_width, precision, width=bar_width, label='Precision', alpha=0.7)
plt.bar(x + 1.5 * bar_width, recall, width=bar_width, label='Recall', alpha=0.7)

# Add labels and title
plt.xlabel('Models')
plt.ylabel('Scores')
plt.title('Model Comparison: Accuracy, F1 Score, Precision, and Recall')
plt.xticks(x, models)
plt.legend()
plt.tight_layout()
plt.show()
```

Figure 12: Plotting the metrics

7. Evaluation

Below metrics were calculated as part of model's performance.

Metrics Calculated:

1. Accuracy
2. F1 Score
3. Precision
4. Recall

```
metrics["Decision Tree"]
{'Accuracy': 0.8033333333333333,
 'F1 Score': 0.593103448275862,
 'Precision': 0.589041095890411,
 'Recall': 0.5972222222222222,
 'Confusion Matrix': array([[198,  30],
        [ 29,  43]], dtype=int64)}

metrics["KNN"]
{'Accuracy': 0.7166666666666667,
 'F1 Score': 0.17475728155339806,
 'Precision': 0.2903225806451613,
 'Recall': 0.125,
 'Confusion Matrix': array([[206,  22],
        [ 63,   9]], dtype=int64)}

metrics["LightGBM"]
{'Accuracy': 0.81,
 'F1 Score': 0.6122448979591837,
 'Precision': 0.6,
 'Recall': 0.625,
 'Confusion Matrix': array([[198,  30],
        [ 27,  45]], dtype=int64)}

metrics["Random Forest"]
{'Accuracy': 0.7933333333333333,
 'F1 Score': 0.44642857142857145,
 'Precision': 0.625,
 'Recall': 0.3472222222222222,
 'Confusion Matrix': array([[213,  15],
        [ 47,  25]], dtype=int64)}

metrics["SVC"]
{'Accuracy': 0.76,
 'F1 Score': 0.0,
 'Precision': 0.0,
 'Recall': 0.0,
 'Confusion Matrix': array([[228,   0],
        [ 72,   0]], dtype=int64)}
```

(1)

```
metrics["Decision Tree"]
{'Accuracy': 0.788,
 'F1 Score': 0.6666666666666666,
 'Precision': 0.6583850931677019,
 'Recall': 0.6751592356687898,
 'Confusion Matrix': array([[1728,  330],
        [ 306,  636]], dtype=int64)}

metrics["KNN"]
{'Accuracy': 0.639,
 'F1 Score': 0.2687373396353815,
 'Precision': 0.3692022263450835,
 'Recall': 0.21125265392781317,
 'Confusion Matrix': array([[1718,  340],
        [ 743,  199]], dtype=int64)}

metrics["LightGBM"]
{'Accuracy': 0.841,
 'F1 Score': 0.7445099089448313,
 'Precision': 0.7513513513513513,
 'Recall': 0.7377919320594479,
 'Confusion Matrix': array([[1828,  230],
        [ 247,  695]], dtype=int64)}

metrics["Random Forest"]
{'Accuracy': 0.8296666666666667,
 'F1 Score': 0.7242309767943875,
 'Precision': 0.7365532381997805,
 'Recall': 0.7123142250530785,
 'Confusion Matrix': array([[1818,  240],
        [ 271,  671]], dtype=int64)}

metrics["SVC"]
{'Accuracy': 0.686,
 'F1 Score': 0.0,
 'Precision': 0.0,
 'Recall': 0.0,
 'Confusion Matrix': array([[2058,    0],
        [ 942,    0]], dtype=int64)}
```

(2)

```
[ ] metrics["Decision Tree"]

{'Accuracy': 0.9993529602070528,
 'F1 Score': 0.99880810488677,
 'Precision': 1.0,
 'Recall': 0.9976190476190476,
 'Confusion Matrix': array([[2251,    0],
        [   2,  838]], dtype=int64)}

[ ] metrics["KNN"]

{'Accuracy': 0.6787447428016823,
 'F1 Score': 0.18806214227309895,
 'Precision': 0.3002610966057441,
 'Recall': 0.13690476190476192,
 'Confusion Matrix': array([[1983,  268],
        [ 725,  115]], dtype=int64)}

[ ] metrics["Random Forest"]

{'Accuracy': 0.999029440310579,
 'F1 Score': 0.998211091234347,
 'Precision': 1.0,
 'Recall': 0.9964285714285714,
 'Confusion Matrix': array([[2251,    0],
        [   3,  837]], dtype=int64)}

[ ] metrics["LightGBM"]

{'Accuracy': 1.0,
 'F1 Score': 1.0,
 'Precision': 1.0,
 'Recall': 1.0,
 'Confusion Matrix': array([[2251,    0],
        [   0,  840]], dtype=int64)}

[ ] metrics["SVC"]

{'Accuracy': 0.7282432869621481,
 'F1 Score': 0.0,
 'Precision': 0.0,
 'Recall': 0.0,
 'Confusion Matrix': array([[2251,    0],
        [ 840,    0]], dtype=int64)}
```

(3)

Figure 13: Results for model 1, 2 and 3 respectively

8. Results and Visualizations

- Light GBM consistently outperformed other models across all datasets.
- For **Insurance Fraud Detection**, it achieved F1 score of 0.612.
- For **Car Insurance Data**, it obtained an accuracy of 0.841 and F1 score of 0.744.
- For **Vehicle Insurance Claim Prediction**, Light GBM achieved perfect scores in all metrics.

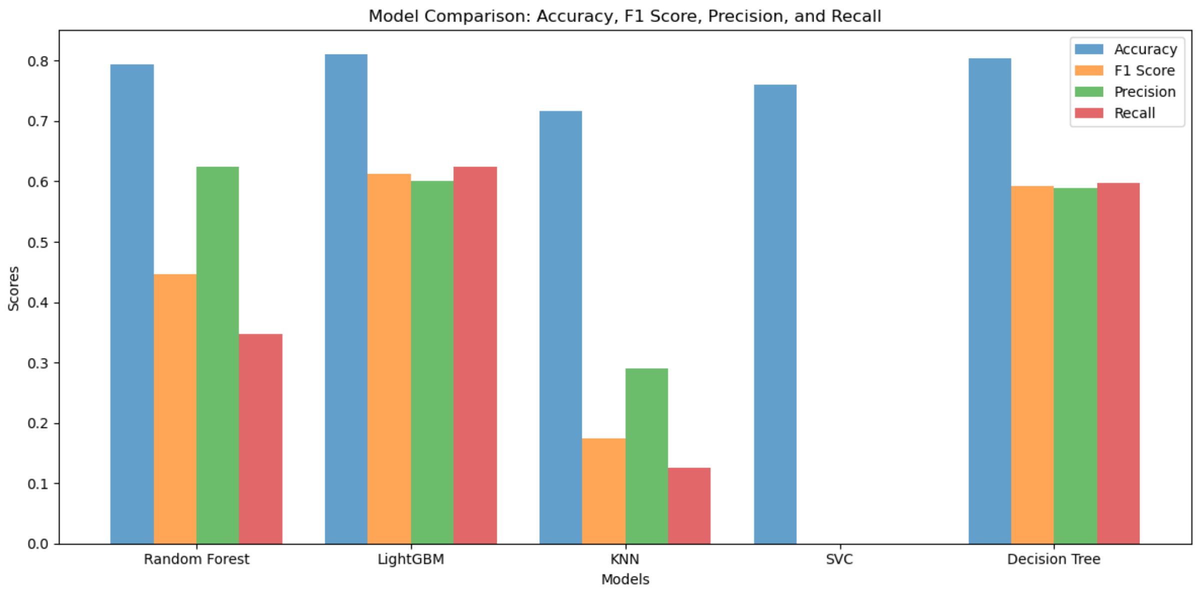

Figure 14: Bar graph for Model comparison (1)

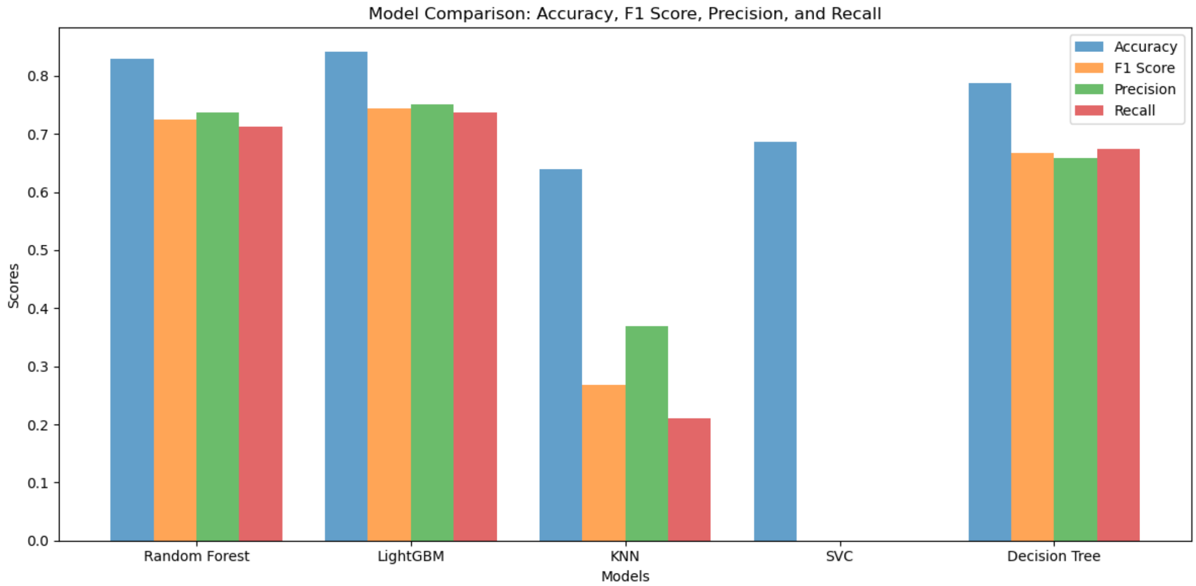

Figure 15: Bar graph for Model comparison (2)

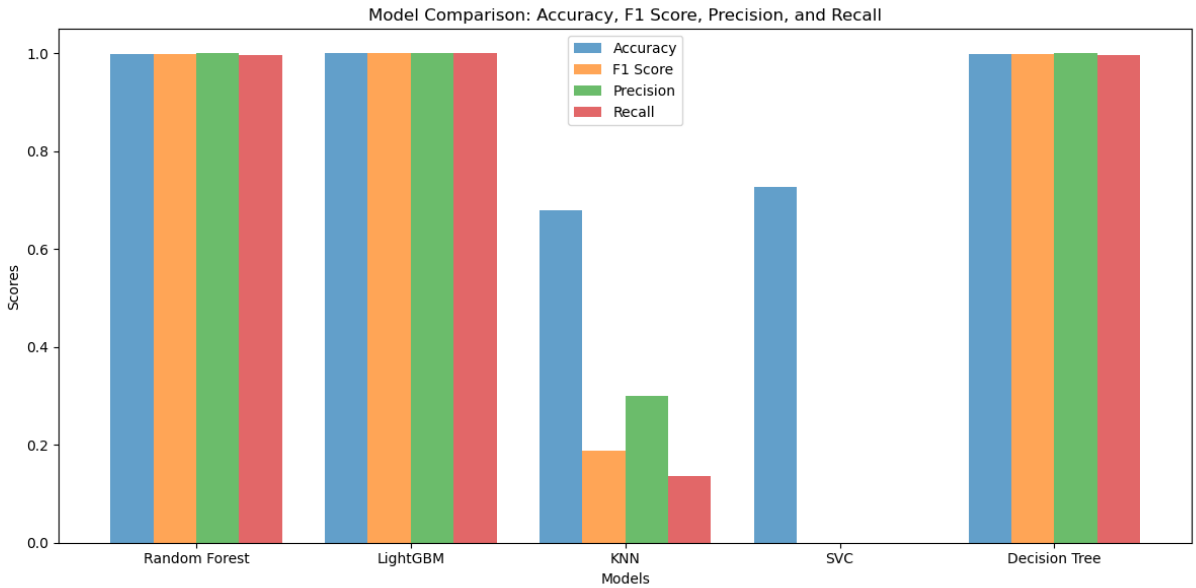

Figure 16: Bar graph for Model comparison (3)