Configuration Manual

MSc Research Project
MSc in Data Analytics

Manish Kumar Mittal
Student ID: x20185596

School of Computing
National College of Ireland

Supervisor: Qurrat Ul Ain

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Manish Kumar Mittal
Student ID:	x20185596
Programme:	MSc in Data Analytics
Year:	2022
Module:	MSc Research Project
Supervisor:	Qurrat Ul Ain
Submission Due Date:	15/08/2022
Project Title:	Configuration Manual
Word Count:	496
Page Count:	10

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:	
Date:	19th September 2022

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

Manish Kumar Mittal
x20185596

1 Introduction

All the requirements for reproducing the research and its outcomes on any individual environment are contained in the configuration manual.This document includes the software and hardware requirements, as well as code for Data Import and Preprocessing, Exploratory Data Analysis, all models built, and Evaluation.

In Section 2, you will find information about the configuration of the environment.The data collection process is described in Section 3.In section 4, data exportation is discussed, including import of libraries, import of datasets, data pre-processing, and exploratory data analysis.Section 5 explains how the training and testing phases are divided and how features are selected.Models, results, and visualizations are described in Section 6.

2 Environment

Detailed information about the hardware and software requirements to implement the research is provided in this section.

2.1 Hardware Requirements

Here are the hardware specifications needed, as shown in Figure 1 and Figure 2.The Apple M1 chip features four performance and four efficiency cores, 8 GB of unified RAM memory, MAC OS 12.3.1, and a 512 GB SSD.

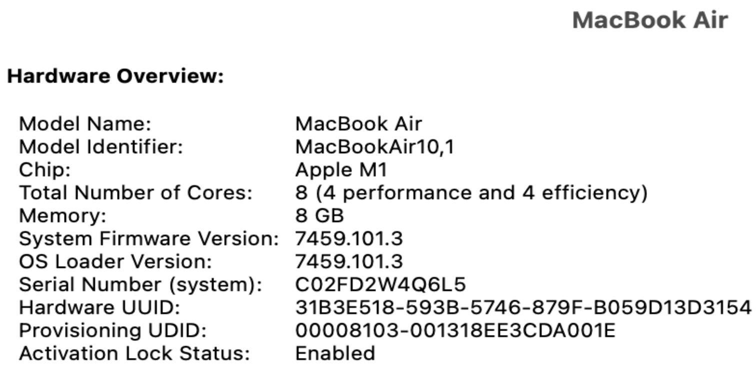

Figure 1: System Hardware Overview

MacBook Air

System Software Overview:

System Version:	macOS 12.3.1 (21E258)
Kernel Version:	Darwin 21.4.0
Boot Volume:	Macintosh HD
Boot Mode:	Normal
Computer Name:	Manish's MacBook Air
Username:	Manish Mittal (manishmittal)
Secure Virtual Memory:	Enabled
System Integrity Protection:	Enabled
Time since boot:	10 days 16:06

Figure 2: System Software Overview

2.2 Software Requirements

- Python (Version 3.7.13)
- Google Colab

3 Data Collection

Kaggle is the source of the dataset.Here is a link to the dataset: `https://www.kaggle.com/datasets/blastchar/telco-customer-churn?sortBy=hotness&group=everyone&pageSize=20&datasetId=13996&language=Python`. The dataset contains 7043 rows and 21 columns.

4 Data Exportation

4.1 Importing Libraries

In Figure 3, you can find all Python libraries you need to implement the entire project.

```
[ ] # Importing packages
    import numpy as np
    import pandas as pd
    import matplotlib.pyplot as plt
    %matplotlib inline
    import seaborn as sns
    import scipy.stats as stats
    from sklearn.preprocessing import LabelEncoder
    from sklearn.model_selection import train_test_split, RandomizedSearchCV
    from sklearn.metrics import recall_score, accuracy_score, classification_report, confusion_matrix, precision_score, f1_score, ConfusionMatrixDisplay
    from sklearn.linear_model import LogisticRegression
    from sklearn.neighbors import KNeighborsClassifier
    from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
    from sklearn.tree import DecisionTreeClassifier
    from imblearn.combine import SMOTEENN
    from sklearn.feature_selection import SelectKBest
    from collections import Counter
    # ignore warning
    import warnings
    warnings.filterwarnings('ignore')
    import matplotlib.ticker as mtick
```

Figure 3: Required Python Libraries

4.2 Importing Dataset

Code to import datasets is shown in Figure 4.

```
from google.colab import files
uploaded = files.upload()
```

Figure 4: Importing Dataset

The code for reading the dataset is shown in Figure 5.

```
#Creating dataframe
data = pd.read_csv('WA_Fn-UseC_-Telco-Customer-Churn.csv')
data.head()
```

Figure 5: Reading Dataset

4.3 Exploratory Data Analysis

A code for visualizing the count of senior citizens can be found in Figure 6.

```
] # pie chart for Count of Senior citizens
  ax = (data['SeniorCitizen'].value_counts()*100.0 /len(data)).plot.pie(autopct='%.1f%%', labels = ['No', 'Yes'],figsize =(5,5), fontsize = 12 )
  ax.yaxis.set_major_formatter(mtick.PercentFormatter())
  ax.set_ylabel('Senior Citizens',fontsize = 12)
  ax.set_title('% of Senior Citizens', fontsize = 12)
```

Figure 6: EDA For Senior Citizen attribute

In Figure 7, the code is shown for visualizing all other attributes of the data.

```
for i, feature in enumerate(categorical_feature):
    if feature != 'TotalCharges':
        if feature != 'customerID':
            plt.figure(i)
            plt.figure(figsize=(12,6))
            sns.countplot(data=data, x=feature, hue='Churn')
plt.show()
```

Figure 7: EDA for All the other attributes

Our target attribute of our data that is churned is represented in Figure 8.

```
# plotting with target feature
sns.countplot(data=data, x='Churn')
plt.title('Count of Churn')
plt.show()
```

Figure 8: EDA for Target attribute

4.4 Data Pre-Processing & Transformation

Based on Figure 9, the only attribute with null values is 'Total Charges'.We replaced these null values with the mean of Total Charges.

```
[ ] #checking null value
    data.isnull().sum()

    customerID           0
    gender               0
    SeniorCitizen        0
    Partner              0
    Dependents           0
    tenure               0
    PhoneService         0
    MultipleLines        0
    InternetService      0
    OnlineSecurity       0
    OnlineBackup         0
    DeviceProtection     0
    TechSupport          0
    StreamingTV          0
    StreamingMovies      0
    Contract             0
    PaperlessBilling     0
    PaymentMethod        0
    MonthlyCharges       0
    TotalCharges        11
    Churn                0
    dtype: int64
```

```
[ ] # replace NaN values with mean value
    data.TotalCharges = data.TotalCharges.fillna(data.TotalCharges.median())
```

Figure 9: Print the Null Values

Data outliers are checked using the code shown in Figure 10.

```
[ ] # Checking for outliers in the data and removing if any

    def remove_outlier(df, col_name):
        plt.figure(figsize=(20,20))
        f, axes = plt.subplots(1, 2,figsize=(12,4))
        sns.boxplot(data = df,x = col_name, ax=axes[0], color='skyblue').set_title("Before Outlier Removal: "+col_name)
        Q1 = df[col_name].quantile(0.25)
        Q3 = df[col_name].quantile(0.75)
        IQR = Q3-Q1
        df[col_name] = df[col_name].apply(lambda x : Q1-1.5*IQR if x < (Q1-1.5*IQR) else (Q3+1.5*IQR if x>(Q3+1.5*IQR) else x))
        sns.boxplot(data = df, x = col_name, ax=axes[1], color='pink').set_title("After Outlier Removal: "+col_name)
        plt.show()
        return df
```

```
[ ] for col in X.columns:
        df = remove_outlier(data,col)
    plt.show()
```

Figure 10: Outliers check

A balanced data set is necessary for better results.Figure 11 depicts the code for balancing the dataset.

```
[ ] #Using smote to balance data
    st=SMOTEENN()
    X_train_st,y_train_st = st.fit_resample(X, y)
    print("The number of classes before fit {}".format(Counter(y)))
    print("The number of classes after fit {}".format(Counter(y_train_st)))

    The number of classes before fit Counter({0: 5174, 1: 1869})
    The number of classes after fit Counter({1: 3123, 0: 2647})
```

Figure 11: Balancing data using SMOTE

5 Data Preparation

5.1 Data Splitting

As shown in Figure 12, below, the training and testing phases are divided 80:20 in the code.

```
[ ] # splitting the over sampling dataset
    X_train_sap, X_test_sap, y_train_sap, y_test_sap = train_test_split(X_train_st, y_train_st, test_size=0.2)
```

Figure 12: Data Split in Train and Test

5.2 Feature Selection

Figure 13 represents the code for selecting the most correlated features.Those features which are considered are denoted as True, while the rest are denoted as False.

```
[ ] # selects the feature which has more correlation
    selection = SelectKBest()  # k=10 default
    X = selection.fit_transform(X,y)

[ ] #this will shows which feature are taken denote as True other are removed like false
    selection.get_support()

    array([False, False, False,  True,  True, False, False, False,  True,
            True,  True,  True, False, False,  True,  True, False,  True,
            True])
```

Figure 13: Feature Selection

Figure 14 and Figure 15 show the code of RandomizedSearchCV for the gradient boosting model, which was used to achieve the best hyperparameters.

```
[ ] gbc_optm = RandomizedSearchCV(estimator=gbc, param_distributions=param_grid,n_iter=100, verbose=3)
    gbc_optm.fit(X_train_sap, y_train_sap)
```

Figure 14: Hyper-parameter Selection for Gradient Boosting

```
[ ] gbc_optm.best_estimator_

    GradientBoostingClassifier(criterion='squared_error', learning_rate=0.5,
                               max_depth=7, max_leaf_nodes=12, min_samples_leaf=11,
                               min_samples_split=5, n_estimators=300)
```

Figure 15: Print Hyper-parameters

6 Model Implementation & Evaluation

All models are implemented and evaluated in this section.

6.1 Logistic Regression

```
[ ] # logistic regression
    Log_reg_sampling = LogisticRegression(C=10, max_iter=150)
    Log_reg_sampling.fit(X_train_sap, y_train_sap)
    Log_sampling_pred = Log_reg_sampling.predict(X_test_sap)

    print(f'Accuracy score : {accuracy_score(Log_sampling_pred, y_test_sap)}')
    print(f'Precision score : {precision_score(Log_sampling_pred, y_test_sap)}')
    print(f'Recall score : {recall_score(Log_sampling_pred, y_test_sap)}')
    print(f'F1 score : {f1_score(Log_sampling_pred, y_test_sap)}')
    print(f'Confusion matrix :\n {confusion_matrix(Log_sampling_pred, y_test_sap)}')
    print(f'Classification report :\n {classification_report(Log_sampling_pred, y_test_sap)}')
    cmd = ConfusionMatrixDisplay(confusion_matrix(Log_sampling_pred, y_test_sap),display_labels=["churn","no churn"])
    cmd.plot()
```

Figure 16: Code for Logistic Regression Model

```
Accuracy score : 0.9199655765920827
Precision score : 0.9294871794871795
Recall score : 0.9220985691573926
F1 score : 0.925778132482043
Confusion matrix :
 [[489  44]
 [ 49 580]]
Classification report :
               precision    recall  f1-score   support

           0       0.91      0.92      0.91       533
           1       0.93      0.92      0.93       629

    accuracy                           0.92      1162
   macro avg       0.92      0.92      0.92      1162
weighted avg       0.92      0.92      0.92      1162

<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7fa3b0dae510>
```

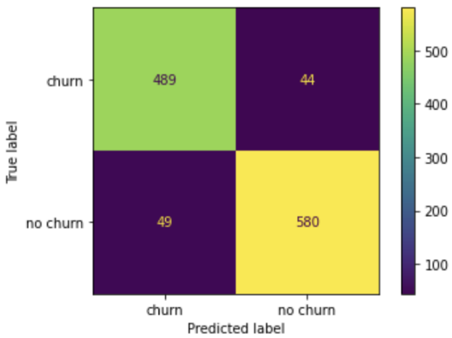

Figure 17: Classification Report and Confusion Matrix for Logistic Regression Model

6.2 Random Forest

```
[ ] # Random forest classifier
    Rfc_sampling = RandomForestClassifier(n_estimators=150,criterion='gini', max_depth=15, min_samples_leaf=10, min_samples_split=6)
    Rfc_sampling.fit(X_train_sap, y_train_sap)
    rfc_sampling_pred = Rfc_sampling.predict(X_test_sap)

    print(f'Accuracy score : {accuracy_score(rfc_sampling_pred, y_test_sap)}')
    print(f'Precision score : {precision_score(rfc_sampling_pred, y_test_sap)}')
    print(f'Recall score : {recall_score(rfc_sampling_pred, y_test_sap)}')
    print(f'F1 score : {f1_score(rfc_sampling_pred, y_test_sap)}')
    print(f'Confusion matrix :\n {confusion_matrix(rfc_sampling_pred, y_test_sap)}')
    print(f'Classification report :\n {classification_report(rfc_sampling_pred, y_test_sap)}')
    cmd = ConfusionMatrixDisplay(confusion_matrix(rfc_sampling_pred, y_test_sap),display_labels=["churn","no churn"])
    cmd.plot()
```

Figure 18: Code for Random Forest Model

```
Accuracy score : 0.9526678141135972
Precision score : 0.9615384615384616
Recall score : 0.9508716323296355
F1 score : 0.9561752988047808
Confusion matrix :
 [[507  24]
 [ 31 600]]
Classification report :
               precision    recall  f1-score   support

           0       0.94      0.95      0.95       531
           1       0.96      0.95      0.96       631

    accuracy                           0.95      1162
   macro avg       0.95      0.95      0.95      1162
weighted avg       0.95      0.95      0.95      1162

<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7fa3b0817810>
```

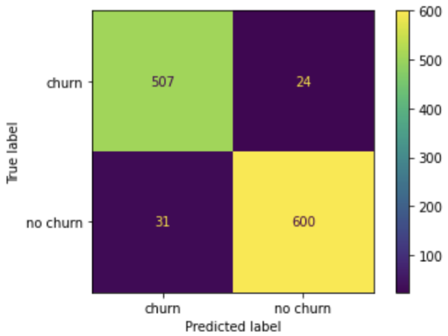

Figure 19: Classification Report and Confusion Matrix for Random Forest Model

6.3 Decision Tree

```
[ ] # decisionTree Classifier
    Dtc_sampling = DecisionTreeClassifier(criterion = "gini",random_state = 100,max_depth=7, min_samples_leaf=15)
    Dtc_sampling.fit(X_train_sap, y_train_sap)
    dtc_sampling_pred = Dtc_sampling.predict(X_test_sap)

    print(f'Accuracy score : {accuracy_score(dtc_sampling_pred, y_test_sap)}')
    print(f'Precision score : {precision_score(dtc_sampling_pred, y_test_sap)}')
    print(f'Recall score : {recall_score(dtc_sampling_pred, y_test_sap)}')
    print(f'F1 score : {f1_score(dtc_sampling_pred, y_test_sap)}')
    print(f'Confusion matrix :\n {confusion_matrix(dtc_sampling_pred, y_test_sap)}')
    print(f'Classification report :\n {classification_report(dtc_sampling_pred, y_test_sap)}')
    cmd = ConfusionMatrixDisplay(confusion_matrix(dtc_sampling_pred, y_test_sap),display_labels=["churn","no churn"])
    cmd.plot()
```

Figure 20: Code for Decision Tree Model

```
[ ]
Accuracy score : 0.9423407917383821
Precision score : 0.9487179487179487
Recall score : 0.9441786283891547
F1 score : 0.9464428457234213
Confusion matrix :
 [[503  32]
 [ 35 592]]
Classification report :
               precision    recall  f1-score   support

           0       0.93      0.94      0.94       535
           1       0.95      0.94      0.95       627

    accuracy                           0.94      1162
   macro avg       0.94      0.94      0.94      1162
weighted avg       0.94      0.94      0.94      1162

<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7fa3ac2876d0>
```

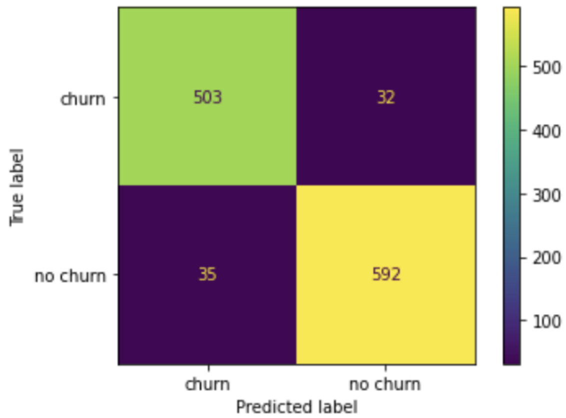

Figure 21: Classification Report and Confusion Matrix for Decision Tree Model

6.4 Gradient Boosting

```
[ ] # GradientBoostingClassifier
gbc_tunning = GradientBoostingClassifier(criterion='squared_error', learning_rate=0.5,
                                         max_depth=7, max_leaf_nodes=12, min_samples_leaf=11,
                                         min_samples_split=5, n_estimators=300)
gbc_tunning.fit(X_train_sap, y_train_sap)
pred = gbc_tunning.predict(X_test_sap)

print(f'Accuracy score : {accuracy_score(pred, y_test_sap)}')
print(f'Precision score : {precision_score(pred, y_test_sap)}')
print(f'Recall score : {recall_score(pred, y_test_sap)}')
print(f'F1 score : {f1_score(pred, y_test_sap)}')
print(f'Confusion matrix :\n {confusion_matrix(pred, y_test_sap)}')
print(f'Classification report :\n {classification_report(pred, y_test_sap)}')
cmd = ConfusionMatrixDisplay(confusion_matrix(pred, y_test_sap),display_labels=["churn","no churn"])
cmd.plot()
```

Figure 22: Code for Gradient Boosting Model

```
Accuracy score : 0.96815834767642
Precision score : 0.9711538461538461
Recall score : 0.9696
F1 score : 0.9703763010408327
Confusion matrix :
 [[519  18]
 [ 19 606]]
Classification report :
               precision    recall  f1-score   support

           0       0.96      0.97      0.97       537
           1       0.97      0.97      0.97       625

    accuracy                           0.97      1162
   macro avg       0.97      0.97      0.97      1162
weighted avg       0.97      0.97      0.97      1162

<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7fa3ac5f4cd0>
```

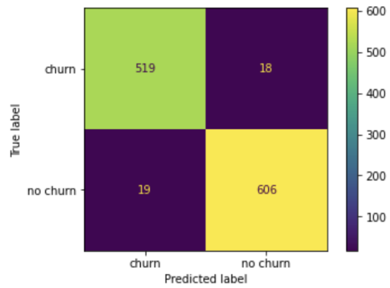

Figure 23: Classification Report and Confusion Matrix for Gradient Boosting Model

6.5 Hybrid Model

```python
#Defining Hybrid Ensemble Learning Model
estimators = []
```

```python
#Defining 5 Decision Tree Classifiers
from sklearn.tree import DecisionTreeClassifier
model1 = DecisionTreeClassifier(max_depth = 3)
estimators.append(('cart1', model1))
model2 = DecisionTreeClassifier(max_depth = 4)
estimators.append(('cart2', model2))
model3 = DecisionTreeClassifier(max_depth = 5)
estimators.append(('cart3', model3))
model4 = DecisionTreeClassifier(max_depth = 2)
estimators.append(('cart4', model4))
model5 = DecisionTreeClassifier(max_depth = 3)
estimators.append(('cart5', model5))
```

```python
#Defining 5 Gradient Boosting classifiers
model11 = GradientBoostingClassifier(max_depth=3)
estimators.append(('gbc1', model11))
model12 = GradientBoostingClassifier(max_depth=4)
estimators.append(('gbc2', model12))
model13 = GradientBoostingClassifier(max_depth=5)
estimators.append(('gbc3', model13))
model14 = GradientBoostingClassifier(max_depth=6)
estimators.append(('gbc4', model14))
model15 = GradientBoostingClassifier(max_depth=7)
estimators.append(('gbc5', model15))
```

```python
#Defining 5 Random Forest Classifiers
model21 = RandomForestClassifier(max_depth = 3,random_state=1)
estimators.append(('rfc1', model21))
model22 = RandomForestClassifier(max_depth = 4,random_state=1)
estimators.append(('rfc2', model22))
model23 = RandomForestClassifier(max_depth = 5,random_state=1)
estimators.append(('rfc3', model23))
model24 = RandomForestClassifier(max_depth = 6,random_state=1)
estimators.append(('rfc4', model24))
model25 = RandomForestClassifier(max_depth = 7,random_state=1)
estimators.append(('rfc5', model25))
```

Figure 24: Creation of 5 instances for Decision Tree Model, Random Forest Model and Gradient Boosting Model

```
[ ] # Defining the ensemble model
    from sklearn.ensemble import VotingClassifier
    ensemble = VotingClassifier(estimators)
    ensemble.fit(X_train_sap, y_train_sap)
    y_pred = ensemble.predict(X_test_sap)
```

```
[ ] def model_performance(modelName,model,X_test_sap,y_test_sap):

        print("Model:",modelName)
        y_pred = model.predict(X_test_sap)
        print("Accuracy Score:",accuracy_score(y_test_sap,y_pred))
        print("Precision Score:",precision_score(y_test_sap,y_pred))
        print("Recall Score:",recall_score(y_test_sap,y_pred))
        print("F1 Score:",f1_score(y_test_sap,y_pred))
        cm = confusion_matrix(y_test_sap,y_pred)
        print("Confusion Matrix:\n",cm)
        cmd = ConfusionMatrixDisplay(cm,display_labels=["churn","no churn"])
        cmd.plot()
        print("Classification Report:\n",classification_report(y_test_sap,y_pred))
        plt.show()
```

```
[ ]  model_performance('HybridModel',ensemble,X_test_sap,y_test_sap)
```

Figure 25: Code for Hybrid Model

```
Model: HybridModel
Accuracy Score: 0.9569707401032702
Precision Score: 0.9526813880126183
Recall Score: 0.967948717948718
F1 Score: 0.9602543720190779
Confusion Matrix:
 [[508  30]
 [ 20 604]]
Classification Report:
               precision    recall  f1-score   support

           0       0.96      0.94      0.95       538
           1       0.95      0.97      0.96       624

    accuracy                           0.96      1162
   macro avg       0.96      0.96      0.96      1162
weighted avg       0.96      0.96      0.96      1162
```

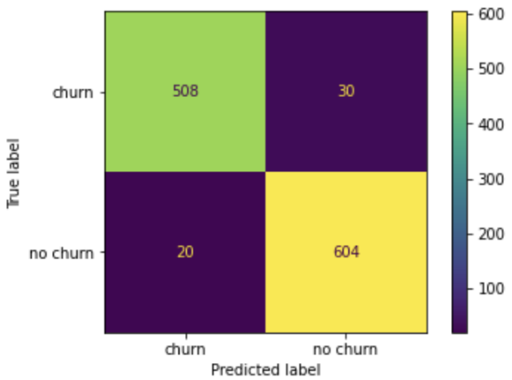

Figure 26: Classification Report and Confusion Matrix for Hybrid Model