National College of Ireland

Configuration Manual

MSc Research Project
MSc Data Analytics

Shreyas Bhargav Bhushan
Student ID: x23175851

School of Computing
National College of Ireland

Supervisor: Vikas Tomer

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Shreyas Bhargav Bhushan
Student ID:	x23175851
Programme:	Msc in Data Analytics
Year:	2024
Module:	MSc Research Project
Supervisor:	Vikas Tomer
Submission Due Date:	29/01/2025
Project Title:	Enhancing Customer Churn Prediction in the Telecom Sector Using Advance Machine Learning Techniques and Explainable AI
Word Count:	1285
Page Count:	11

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:	29th January 2025

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 keepa 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

Shreyas Bhargav Bhushan
Student ID: x23175851

1 Introduction

Customer Churn affects the profits of the telecom company, as it is cheaper to maintain the customer than acquiring new one. This research proposes heterogeneous multi-stacking with Random Forest, XGBoost, KNN, and Logistic regression and implementing techniques like SMOTE, RFE and SHAP to improve performance and interpretability of the proposed model. These insights enable the companies to develop personalised retention strategies and improving stakeholders trust and business.

This document contains all the necessary configuration that would be required for reproducing the outcome of the code. The document discusses the software and hardware requirements, importing the required libraries, step by step process for preparing the data which includes data cleaning, feature selection, and handling of imbalance class, then the development of multi-stack model and application of SHAP.

2 Environment Setup

This section contains the detail information about the software and hardware requirements that would be required to execute the code.

2.1 Hardware Requirements:

Fig.1 Configuration of the system used in the research

The Fig.1 shows that the research made use of Lenovo Legion 5 15IMH05 alptop equipped with an Intel core i5-10300H CPU at 2.50 GHz, a 64-bit operating with a RAM of 8GB.

2.2 Software Requirements:

Operating System Windows:

- Windows 10 or later
- macOS: macOS 10.14 or later
- Linux: Ubuntu 20.04 or later

Development Environment:

- JupyterNotebook
- Google Colab

Python Version:

- Python 3.9.

Anaconda: For managing Python environments and packages.

3 Data Collection

In this study we have made use of publicly available IBM Telco Customer Churn Dataset from Kaggle which contains 7043 rows and 21 features which are divided into 3 different categories: service-related, account-related and demographic-related. The link of the dataset is given below.

The Link for IBM Telco Customer Churn Dataset from Kaggle:
https://www.kaggle.com/datasets/blastchar/telco-customer-churn/data

4 Importing Libraries

The Fig.2 shows the importing of necessary libraries that is required for data preprocessing, data visualization and Machine learning the project uses libraries such as Scikit learn, Xgboost, Catboost for model building and model evaluation and for data manipulation and analysis the project uses libraries such as pandas, numpy and missingno. Visualization is facilitated by the library such as matplotlib, seaborn and plotly for generating better graphics. Used models are Decision Tree, Random Forest, Naive Bayes, SVM, KNN, Logistic Regression Boosting algorithms.

```
import pandas as pd
import numpy as np
import missingno as msno
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import plotly.graph_objects as go
from plotly.subplots import make_subplots
import warnings
warnings.filterwarnings('ignore')

from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import LabelEncoder

from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.neural_network import MLPClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from xgboost import XGBClassifier
from catboost import CatBoostClassifier
from sklearn import metrics
from sklearn.metrics import roc_curve
from sklearn.metrics import recall_score, confusion_matrix, precision_score, f1_score, accuracy_score, classification_report
```

Fig.2 Importing Libraries

5 Exploratory Data Analysis

The exploration of the dataset helps us to understand the importance of different features that are influencing the customer churn. Following figure shows the code snippets for some of the data visualization.

```
# Payment Method Distribution with respect to Churn
payment_churn_distribution = df.groupby(['PaymentMethod', 'Churn']).size().unstack()

# Plotting the distribution
payment_churn_distribution.plot(kind='bar', figsize=(12, 6), alpha=0.8)
plt.title('Customer Payment Method Distribution with respect to Churn')
plt.xlabel('Payment Method')
plt.ylabel('Count')
plt.xticks(rotation=45, ha='right')
plt.legend(title='Churn', labels=['No', 'Yes'])
plt.show()
```

Fig.3 Payment Distribution with respect to Churn

The Fig.3 presents a code to understand how the payment method is distrbuted among the customer who churned and those who did not.

```
# Churn distribution with respect to Internet Service and Gender
churn_internet_gender_distribution = df.groupby(['gender', 'InternetService', 'Churn']).size().unstack()

# Plotting the distribution
churn_internet_gender_distribution.plot(kind='bar', figsize=(14, 7), alpha=0.8)
plt.title('Churn Distribution with respect to Internet Service and Gender')
plt.xlabel('Gender and Internet Service')
plt.ylabel('Count')
plt.xticks(rotation=45, ha='right')
plt.legend(title='Churn', labels=['No', 'Yes'])
plt.show()
```

Fig.4 Churn Distribution with repsct to Internet Service and Gender

The code in the Fig.4 helps us to bar graph to under the churn distribution with respect to interservice type and gender.

```
# Churn distribution with respect to Partners
churn_partner_distribution = df.groupby(['Partner', 'Churn']).size().unstack()

# Plotting the distribution
churn_partner_distribution.plot(kind='bar', figsize=(10, 6), alpha=0.8)
plt.title('Churn Distribution with respect to Partners')
plt.xlabel('Partner Status')
plt.ylabel('Count')
plt.xticks(rotation=0)
plt.legend(title='Churn', labels=['No', 'Yes'])
plt.show()
```

Fig.5 Churn Distribution with respect to Partners

The Fig.5, shows the distribution of churn with repesct to the relationship status of the customers.

6 Data Preparation

```
# Visualize missing values as a matrix
msno.matrix(df);
```

Fig.6 Checking for missing values

The Fig.6 shows that we use missingo library to visualize missing values in a dataset as matrix plot. The Fig.7 shows the code to the inderect missingness that may be present in the feature ToatalCharges.

```
#There are some indirect missingness in our data.
df['TotalCharges'] = pd.to_numeric(df.TotalCharges, errors='coerce')
df.isnull().sum()
```

Fig. 7 Checking possible missingness in our data

```
# Identify potential outliers using boxplots for numerical columns
numerical_cols = ['tenure', 'MonthlyCharges', 'TotalCharges']

# Create boxplots for each numerical column
for col in numerical_cols:
    plt.figure(figsize=(10, 6))
    sns.boxplot(data=df, x=col, palette='Set2')
    plt.title(f'Boxplot for {col}')
    plt.xlabel(col)
    plt.show()
```

Fig. 8 Checking for Outliers

The Fig.8, displays the code for plotting boxplot in order to check for the presence of outliers in the numerical columns.

```
from sklearn.feature_selection import RFE
from sklearn.ensemble import RandomForestClassifier

# Initialize a RandomForestClassifier for RFE
model = RandomForestClassifier(random_state=40)
rfe = RFE(estimator=model, n_features_to_select=10)  # Selecting the top 10 features

# Fit RFE on the training data
rfe.fit(X_train, y_train)

# Identify the selected features
rfe_selected_features = X_train.columns[rfe.support_]
rfe_feature_ranking = pd.Series(rfe.ranking_, index=X_train.columns).sort_values()

# Display the selected features and their rankings
selected_features_rfe = pd.DataFrame({
    'Feature': X_train.columns,
    'Rank': rfe.ranking_
}).sort_values(by="Rank")

# Display the selected features and their rankings
print(selected_features_rfe)
```

Fig.9 Feature slection using RFE

The Fig.9, provides code for the slection of the 10 feature that plays significant role in customer churn using Recursive Feature Elimination technique.

```python
from imblearn.over_sampling import SMOTE

# Initialize SMOTE for oversampling
smote = SMOTE(random_state=40)

# Apply SMOTE to the training data
X_train_resampled, y_train_resampled = smote.fit_resample(X_train, y_train)

# Check the new class distribution after SMOTE
resampled_class_distribution = pd.Series(y_train_resampled).value_counts(normalize=True) * 100

# Display the new class distribution as a bar chart
plt.figure(figsize=(8, 5))
resampled_class_distribution.plot(kind='bar', color=['skyblue', 'orange'])
plt.title('Class Distribution after SMOTE')
plt.xlabel('Class (Churn)')
plt.ylabel('Percentage')
plt.xticks([0, 1], labels=['No Churn', 'Churn'], rotation=0)
plt.show()

# Display the actual distribution percentages
resampled_class_distribution
```

Fig. 10 Application of SMOTE Technique

The Fig.10, depicts the code snippet for the apllication of Synthetic Minority Oversampling Technique to handle the class imbalance.

7 Model Development and Interpretability

7.1 Random Forest Model

```python
# Define the parameter grid for Random Forest
param_grid = {
    'n_estimators': [50, 100, 200],
    'max_depth': [10, 20, 30, None],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'bootstrap': [True, False]
}

# Initialize Random Forest Classifier
rf_model = RandomForestClassifier(random_state=42)

# Perform GridSearchCV with 5-fold cross-validation
grid_search = GridSearchCV(estimator=rf_model,
                           param_grid=param_grid,
                           scoring='accuracy',
                           cv=5,
                           n_jobs=-1,
                           verbose=2)

# Fit the model on the resampled training data
grid_search.fit(X_train_resampled, y_train_resampled)
```

Fig. 11: Code for Hyperparameter Tuning using GridSearchCV of Random Forest

```python
import shap
import matplotlib.pyplot as plt

# Initialize the SHAP explainer
explainer = shap.TreeExplainer(best_rf_model)

# Calculate SHAP values for the test data
shap_values = explainer.shap_values(X_test)

# Summary plot for feature importance
shap.summary_plot(shap_values[1], X_test, plot_type="bar", show=False)
plt.title("SHAP Feature Importance (Bar Plot)")
plt.show()

# SHAP force plot for individual predictions
sample_index = 0  # Select an example
shap.force_plot(explainer.expected_value[1], shap_values[1][sample_index], X_test.iloc[sample_index], matplotlib=True)
```

Fig. 12: SHAP analysis for Random Forest model

7.2 XGBoost Model

```python
# Define parameter grid with additional parameters
param_grid = {
    'n_estimators': [100, 200, 300],
    'learning_rate': [0.01, 0.05, 0.1],
    'max_depth': [3, 6, 10],
    'min_child_weight': [1, 3, 5],
    'subsample': [0.8, 1.0],
    'colsample_bytree': [0.8, 1.0],
    'gamma': [0, 0.1, 0.2],  # Regularization term
    'scale_pos_weight': [1, 3, 5],  # For imbalanced classes
    'max_delta_step': [0, 1, 5]  # Helps with imbalance
}

# Initialize XGBoost model
xgb_model = XGBClassifier(random_state=40)

# Initialize RandomizedSearchCV
random_search = RandomizedSearchCV(estimator=xgb_model, param_distributions=param_grid,
                                   n_iter=100, scoring='accuracy', cv=5, n_jobs=-1, verbose=2, random_state=40)

# Fit the RandomizedSearchCV
random_search.fit(X_train_resampled, y_train_resampled)

# Best hyperparameters found by RandomizedSearchCV
print("Best Parameters Found by RandomizedSearchCV:")
print(random_search.best_params_)
```

Fig. 13: Code for Hyperparameter Tuning using RandomizedSearchCV XGBoost model

```python
# SHAP for XGBoost model
# Initialize the SHAP explainer for XGBoost model (tree-based)
explainer_xgb = shap.TreeExplainer(best_xgb_model)
# Calculate SHAP values for the test data
shap_values_xgb = explainer_xgb.shap_values(X_test)
# Summary plot for feature importance (bar plot)
shap.summary_plot(shap_values_xgb, X_test, plot_type="bar", show=False)
plt.title("SHAP Feature Importance (Bar Plot) - XGBoost")
plt.show()

# SHAP force plot for an individual prediction
sample_index_xgb = 0  # Select an example from test data

# For binary classification, shap_values_xgb is a single array (not a list)
shap_values_class = shap_values_xgb  # No need to index

# Create the SHAP Explanation object for the selected instance
shap_force_data = shap.Explanation(values=shap_values_class[sample_index_xgb],
                                   base_values=explainer_xgb.expected_value,
                                   data=X_test.iloc[sample_index_xgb],
                                   feature_names=X_test.columns)
# Plot the SHAP force plot
shap.force_plot(shap_force_data)
```

Fig.14 SHAP analysis for XGBoost Model

7.3 k-nearest neighbors (KNN)

```python
# Define the expanded hyperparameter distribution for RandomizedSearchCV
param_dist = {
    'n_neighbors': np.arange(3, 21, 2),  # Search for odd values between 3 and 21
    'weights': ['uniform', 'distance'],
    'algorithm': ['auto', 'ball_tree', 'kd_tree', 'brute'],
    'p': [1, 2],  # p=1 is Manhattan distance, p=2 is Euclidean distance
    'leaf_size': np.arange(10, 60, 10),  # Range for leaf_size parameter
    'metric': ['minkowski', 'euclidean', 'manhattan', 'chebyshev'],  # Different distance metric
}

# Initialize KNN model
knn = KNeighborsClassifier()

# Initialize RandomizedSearchCV with 10-fold cross-validation
random_search = RandomizedSearchCV(estimator=knn, param_distributions=param_dist,
                                   n_iter=100, cv=10, scoring='accuracy',
                                   random_state=42, n_jobs=-1, verbose=2)

# Perform RandomizedSearchCV to find the best hyperparameters
random_search.fit(X_train_scaled, y_train)

# Best hyperparameters found by RandomizedSearchCV
print("Best Hyperparameters from RandomizedSearchCV: ", random_search.best_params_)

# Get the best KNN model from RandomizedSearchCV
best_knn_model = random_search.best_estimator_
```

Fig.15: Code for Hyperparameter Tuning using RandomizedSearchCV XGBoost model

```python
# SHAP for KNN model using KernelExplainer
background = shap.sample(X_train, 100)

# Initialize KernelExplainer for KNN (non-tree-based model)
explainer_knn = shap.KernelExplainer(best_knn_model.predict_proba, background)

# Calculate SHAP values for the test data (subsample to speed up if needed)
shap_values_knn = explainer_knn.shap_values(X_test[:500])

# Summary plot for feature importance
shap.summary_plot(shap_values_knn, X_test[:500], feature_names=X.columns, plot_type="bar", show=False)
plt.title("SHAP Feature Importance (Bar Plot) - KNN")
plt.show()
```

Fig.16: SHAP analysis for KNN

7.4 Multi-stacked Model

```
# Initialize the base models
knn_model = KNeighborsClassifier()
rf_model = RandomForestClassifier(random_state=42)
xgb_model = XGBClassifier(random_state=42)

# Perform RandomizedSearchCV for KNN
knn_search = RandomizedSearchCV(knn_model, knn_param_grid, n_iter=10, cv=5, scoring='accuracy', random_state=42, n_jobs=-1, verbose=2)
knn_search.fit(X_train_scaled, y_train)

# Perform RandomizedSearchCV for Random Forest
rf_search = RandomizedSearchCV(rf_model, rf_param_grid, n_iter=10, cv=5, scoring='accuracy', random_state=42, n_jobs=-1, verbose=2)
rf_search.fit(X_train_scaled, y_train)

# Perform RandomizedSearchCV for XGBoost
xgb_search = RandomizedSearchCV(xgb_model, xgb_param_grid, n_iter=10, cv=5, scoring='accuracy', random_state=42, n_jobs=-1, verbose=2)
xgb_search.fit(X_train_scaled, y_train)
```

```
# Create the new Stacked Model using the best base models and the meta-model
base_learners = [
    ('knn', best_knn_model),
    ('rf', best_rf_model),
    ('xgb', best_xgb_model)
]

# Initialize the meta-model (Logistic Regression)
meta_model = best_lr_model

# Create the Stacked model
stack_model = StackingClassifier(estimators=base_learners, final_estimator=meta_model)

stack_model.fit(X_train_scaled, y_train)

y_pred = stack_model.predict(X_test_scaled)
y_prob = stack_model.predict_proba(X_test_scaled)[:, 1]  # For ROC-AUC
```

Fig.15: Code for developing Multi-stacked model

```
# Explain the predictions of the stacked model
class StackedModelWrapper:
    def __init__(self, model):
        self.model = model

    def predict_proba(self, X):
        return self.model.predict_proba(X)

# Wrap the stacked model
stacked_model_wrapper = StackedModelWrapper(stack_model)
# Create a SHAP explainer using KernelExplainer for stacked model
explainer = shap.KernelExplainer(stacked_model_wrapper.predict_proba, X_train_scaled[:100])
# Compute SHAP values for the test data
shap_values = explainer.shap_values(X_test_scaled)
# Visualize global feature importance
shap.summary_plot(shap_values[1], X_test_scaled, feature_names=X.columns)
# Visualize local explanations for a single instance
index_to_explain = 0
shap.force_plot(explainer.expected_value[1], shap_values[1][index_to_explain], X_test_scaled[index_to_explain], feature_names=X.columns)
```

Fig.16: SHAP analysis for Multi-stacked Model

8 Conclusion

In conclusion this document provides the details information on how to set up the environment, list the necessary software and hardware requirements, how to reproduce the results by performing data preparation, developing the machine learning models including the multi-stacked ensemble model and application of SHAP for interpretability. With the help of this document the reader can easily perform the experiments and reproduce the results.

References

Aggarwal, P. and Vijayakumar, V., 2024, May. Customer Churn Prediction in the Telecom Sector. In *2024 3rd International Conference on Artificial Intelligence For Internet of Things (AIIoT)* (pp. 1-6). IEEE.

Ahmad, A.K., Jafar, A. and Aljoumaa, K., 2019. Customer churn prediction in telecom using machine learning in big data platform. *Journal of Big Data*, *6*(1), pp.1-24.

Barsotti, A., Gianini, G., Mio, C., Lin, J., Babbar, H., Singh, A., Taher, F. and Damiani, E., 2024. A Decade of Churn Prediction Techniques in the TelCo Domain: A Survey. *SN Computer Science*, *5*(4), p.404.

Bharambe, Y., Deshmukh, P., Karanjawane, P., Chaudhari, D. and Ranjan, N.M., 2023, January. Churn prediction in telecommunication industry. In *2023 International Conference for Advancement in Technology (ICONAT)* (pp. 1-5). IEEE.

Chang, V., Hall, K., Xu, Q.A., Amao, F.O., Ganatra, M.A. and Benson, V., 2024. Prediction of Customer Churn Behavior in the Telecommunication Industry Using Machine Learning Models. *Algorithms*, *17*(6), p.231.

Faraji Googerdchi, K., Asadi, S. and Jafari, S.M., 2024. Customer churn modeling in telecommunication using a novel multi-objective evolutionary clustering-based ensemble learning. *Plos one*, *19*(6), p.e0303881.

He, C. and Ding, C.H., 2024. A novel classification algorithm for customer churn prediction based on hybrid Ensemble-Fusion model. *Scientific Reports*, *14*(1), p.20179.

Kavitha, C., Tripathi, G., Sridivya, R., Yamuna, V., Supriya, N. and Lakshmanarao, A., 2024, April. An Efficient Churn Prediction model using ML supervised and semi supervised Learning Techniques. In *2024 IEEE 9th International Conference for Convergence in Technology (I2CT)* (pp. 1-6). IEEE.

Kirgiz, O.B., Kiygi-Calli, M., Cagliyor, S. and El Oraiby, M., 2024. Assessing the effectiveness of OTT services, branded apps, and gamified loyalty giveaways on mobile customer churn in the telecom industry: A machine-learning approach. *Telecommunications Policy*, *48*(8), p.102816.

Ouf, S., Mahmoud, K.T. and Abdel-Fattah, M.A., 2024. A proposed hybrid framework to improve the accuracy of customer churn prediction in telecom industry. *Journal of Big Data*, *11*(1), p.70.

Poudel, S.S., Pokharel, S. and Timilsina, M., 2024. Explaining customer churn prediction in telecom industry using tabular machine learning models. *Machine Learning with Applications*, *17*, p.100567.

Shaikhsurab, M. a. M. P., 2024. Enhancing Customer Churn Prediction in Telecommunications: An Adaptive Ensemble Learning Approach. *arXiv preprint*, Volume arXiv:2408.16284.

Sharma, A., Tyagi, M. and Kumar, S., 2023, June. Telecom Churn Prediction Using Python. In *International Conference on Recent Trends in Computing* (pp. 423-434). Singapore: Springer Nature Singapore.

Sikri, A., Jameel, R., Idrees, S.M. and Kaur, H., 2024. Enhancing customer retention in telecom industry with machine learning driven churn prediction. *Scientific Reports*, *14*(1), p.13097.

Usman-Hamza, F.E., Balogun, A.O., Amosa, R.T., Capretz, L.F., Mojeed, H.A., Salihu, S.A., Akintola, A.G. and Mabayoje, M.A., 2024. Sampling-based novel heterogeneous multi-layer stacking ensemble method for telecom customer churn prediction. *Scientific African*, *24*, p.e02223.

Wagh, S.K., Andhale, A.A., Wagh, K.S., Pansare, J.R., Ambadekar, S.P. and Gawande, S.H., 2024. Customer churn prediction in telecom sector using machine learning techniques. *Results in Control and Optimization*, *14*, p.100342.

Wang, C., Rao, C., Hu, F., Xiao, X. and Goh, M., 2024. Risk assessment of customer churn in telco using FCLCNN-LSTM model. *Expert Systems with Applications*, *248*, p.123352.