Configuration Manual

MSc Research Project
Msc Fintech

Dhanush Raju
Student ID: x22196757

School of Computing
National College of Ireland

Supervisor: Sean Heany
& Noel Cosgrave

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Dhanush Raju		
Student ID:	X22196757		
Programme:	MSc Fintech	**Year:**	2023 - 2024
Module:	Practicum		
Lecturer:	Sean Heeney & Noel Cosgrave		
Submission Due Date:	12/08/2024		
Project Title:	Identifying the probability of the Natural Disaster to help the Insurance Company to take decisions on providing Insurances		
Word Count:		**Page Count:**	

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: Dhanush Raju ……………………………………………………………………………………………………

Date: 12/08/2024 ……………………………………………………………………………………………………

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

Dhanush Raju
Student ID:x22196757

1 Introduction

This Configuration manual provides the setup, requirements and steps by step process of the codes for the topic “Identifying the probability of the Natural Disaster to help the Insurance Company to take decisions on providing Insurances”.

2 System Requirements

2.1 Hardware

- RAM: 8GB DDR3
- OS: Windows 11 pro
- Processor: i5 9th generation

2.2 Software

- Google Colab Notebook
- Python

3 Data Collection

The below datasets are used for the study

The First dataset can be accessed at https://ourworldindata.org/natural-disasters. This dataset provides information on worldwide instances of natural disasters and their economic aftermath. Natural disasters addressed include drought, earthquakes, floods, impacts, extreme weather, and volcanic activity. Information about the cumulative incidence of multiple catastrophic catastrophes is also included.

The Second dataset can be https://data.mendeley.com/datasets. It contains the information on the client together with the specifics of the insurance policy. In addition to this, it contains the information pertaining to the incident that led to the filing of the claims. The dataset that has been provided to us has a total of 1000 rows and 40 columns. The titles of the columns, such as the policy number, the policy bind date, the yearly premium, the severity of the event, the location of the occurrence, the vehicle model, and so on.

4 Installing Packages

Python Libraries that are used in the project

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Figure 1 Import Libraries

```
from sklearn.model_selection import StratifiedKFold
```

```
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.decomposition import PCA
from sklearn.pipeline import Pipeline
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score,classification_report,confusion_matrix,ConfusionMatrixDisplay
from sklearn.naive_bayes import GaussianNB
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
```

Figure 2: Import Machine learning models

Importing the required libraries from Pandas, numpy, matplotlib and sklearn.ensemble, sklearn. naive_bayes, sklearn.linear_model, respectively, to start the machine learning workflow: RandomForestClassifier, GradientBoostingClassifier, GaussianNB, and LogisticRegression for building different classifiers. Use sklearn.decomposition to import PCA for dimensionality reduction. Pipeline from sklearn.pipeline can be used to automate several workflow steps. Import train_test_split from sklearn.model_selection to split the dataset for model selection and assessment. Import accuracy_score, classification_report, confusion_matrix, and ConfusionMatrixDisplay from sklearn.metrics to evaluate the performance of the model. Finally, import StandardScaler for feature scaling from sklearn.preprocessing.

Data Cleaning and Preprocessing

```
df=df.drop(['declaration_date','last_refresh','hm_program_declared', 'incident_begin_date',
       'incident_end_date', 'disaster_closeout_date', 'fips', 'place_code',
       'designated_area', 'declaration_request_number','hash','id'],axis=1)
```

Figure 3: The code removes a set of specified columns from the DataFrame

```python
X=df_encoded.drop(["declaration_type",'declaration_title'],axis=1)
y=df["declaration_type"]
```

Figure 4: This code splits the DataFrame into features (X) and target variable (y).

```python
classifiers={
    'RandomForestClassifier':RandomForestClassifier(),
    'GradientBoostingClassifier':GradientBoostingClassifier(),
    'GaussianNB':GaussianNB(),
    'LogisticRegression':LogisticRegression(),
    'SVM-sigmoid':SVC(kernel='sigmoid'),
    'SVM-rbf':SVC(kernel='rbf'),
    # 'SVM-linear':SVC(kernel='linear'),
     'LGBMClassifier':LGBMClassifier(),
    #  'XGBClassifier':XGBClassifier()
}
```

Figure 5: Creates a dictionary of classifiers, each associated with a different machine learning algorithm or model, including RandomForest, GradientBoosting, Naive Bayes, Logistic Regression, Support Vector Machines with different kernels, LightGBM, and XGBoost

```python
for clf_name,clf in classifiers.items():
  print(f"Training and Evaluating {clf_name}")
  clf.fit(X_train,y_train)
  y_pred=clf.predict(X_test)
  print(f"{clf_name}Accuracy: {accuracy_score(y_test,y_pred)}")
  print(classification_report(y_test,y_pred))
  cm=confusion_matrix(y_test,y_pred)
  disp=ConfusionMatrixDisplay(confusion_matrix=cm,display_labels=clf.classes_)
  disp.plot()
  plt.show()
```

Figure 6: This code trains and evaluates each classifier in the `classifiers` dictionary. It prints the accuracy, classification report, and confusion matrix for each model, and then displays the confusion matrix as a plot.

```python
# Libraries for exploring, handling and visualizing data
import pandas as pd, numpy as np, matplotlib.pyplot as plt, seaborn as sns, plotly.express as px
# Sklearn's preprocessing library
from sklearn.preprocessing import StandardScaler
# Importing train and test data split
from sklearn.model_selection import train_test_split
# Sklearn's metrics to evaluate our models
from sklearn.metrics import accuracy_score, precision_score, confusion_matrix, recall_score, f1_score
# Classifiers
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier

# Setting theme style and color palette to seaborn
sns.set_theme(context = 'notebook', style='darkgrid',palette='dark')
```

Figure 7: This code imports libraries for data handling, visualization, model evaluation, and machine learning, and sets a theme for Seaborn visualizations.

.

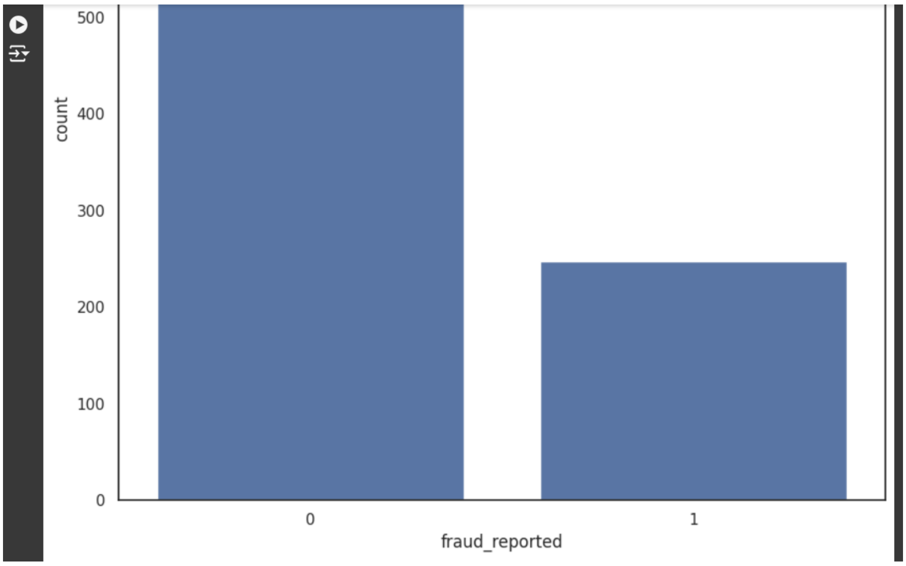

Figure 8 : Shows the fraud report of yes or no from the Insurance dataset

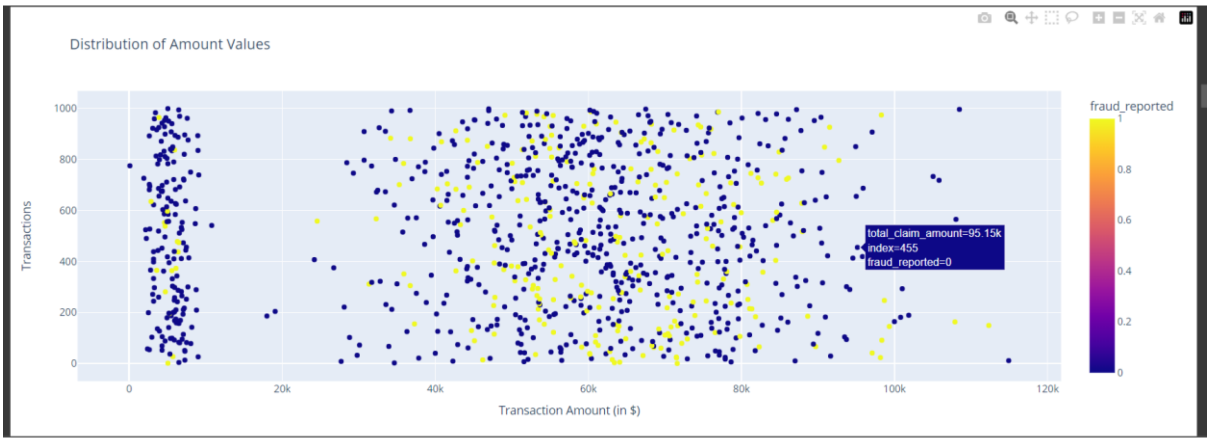

Figure 9 : It displays the distribution of Amount values in fraud report.

```python
from sklearn.linear_model import LogisticRegression
import xgboost as xgb
# Preparing Classifiers
decision_tree = DecisionTreeClassifier()
random_forest = RandomForestClassifier(n_estimators=100)
logistic_regression = LogisticRegression(random_state=0)
params = {
    'min_child_weight': 1,
    'gamma': 0.5,
    'subsample': 0.8,
    'colsample_bytree': 0.8,
    'max_depth': 5
    }
xg_boost = xgb.XGBClassifier(min_child_weight=params['min_child_weight'],
                    gamma=params["gamma"],
                    subsample=params["subsample"],
                    colsample_bytree=params['colsample_bytree'],
                    max_depth=params["max_depth"])

# Decision Tree
decision_tree.fit(train_x,train_y)
predictions_dt = decision_tree.predict(test_x)
decision_tree_score = round(decision_tree.score(test_x,test_y) * 100, 2)

# Random Forest
random_forest.fit(train_x,train_y)
prediction_rf = random_forest.predict(test_x)
random_forest_score = round(random_forest.score(test_x,test_y) * 100,2)

# logistic Regression
logistic_regression.fit(train_x,train_y)
prediction_lr = logistic_regression.predict(test_x)
logistic_regression_score = round(logistic_regression.score(test_x,test_y) * 100,2)

# Xg Boost
xg_boost.fit(train_x,train_y)
prediction_xgb = xg_boost.predict(test_x)
xg_boost_score = round(xg_boost.score(test_x,test_y) * 100,2)

print('Decision Tree Performance: ', decision_tree_score)
print('Random Forest Performance: ', random_forest_score)
print('Logistic Regression Performance: ', logistic_regression_score)
print('XgBoost Performance: ', xg_boost_score)
```
Figure 10: This code trains and evaluates four classifiers:

```python
# Confusion Matrix
confusion_matrix_decision_tree = confusion_matrix(test_y, predictions_dt)
# Visualization
ax = plt.subplot()
sns.heatmap(confusion_matrix_decision_tree, annot=True, fmt='g', ax = ax)
ax.set_xlabel('Predicted Values')
ax.set_ylabel('Actual Values')
ax.set_title('Confusion Matrix - Decision Tree')
ax.xaxis.set_ticklabels(['Genuine','Fraud'])
ax.yaxis.set_ticklabels(['Genuine','Fraud'])
```

Figure 11: This code generates and visualizes a confusion matrix for the Decision Tree classifier. It uses a heatmap to display the matrix with labels for predicted and actual values.

```python
# Importing SMOTE from imblearn lib
from imblearn.over_sampling import SMOTE
resampled_x, resampled_y = SMOTE().fit_resample(X,Y) # reshaping data

print('X New Shape: ', resampled_x.shape)
print('Y New Shape: ', resampled_y.shape)
```

Figure 12: This code imports `SMOTE` from the `imblearn` library and applies it to the dataset. `SMOTE` (Synthetic Minority Over-sampling Technique) generates synthetic samples to balance the class distribution in the dataset. The shapes of the resampled feature set (`resampled_x`) and labels (`resampled_y`) are then printed to show the new dimensions after resampling.

```python
# Splitting our resampled data
(train_x,test_x,train_y,test_y) = train_test_split(resampled_x, resampled_y,
                                                    test_size = 0.3, random_state = 42)

print("train_x size: ", train_x.shape)
print("test_x size: ", test_x.shape)
```

Figure 13: This code splits the resampled data into training and testing sets.
The `train_test_split` function is used to divide `resampled_x` and `resampled_y` into training and test datasets with 30% of the data reserved for testing. The shapes of the training and testing datasets are then printed to confirm the split.

```python
# Random Forest
random_forest = RandomForestClassifier(n_estimators= clf_rfc.best_params_['n_estimators'], max_features=clf_dtc.best_params_['max_features'])
random_forest.fit(train_x,train_y)
prediction_rf = random_forest.predict(test_x)
random_forest_score = round(random_forest.score(test_x,test_y) * 100,2)

# logistic Regression
logistic_regression = LogisticRegression(random_state=0,penalty = clf_lr.best_params_['penalty'],C = clf_lr.best_params_['C'])
logistic_regression.fit(train_x,train_y)
prediction_lr = logistic_regression.predict(test_x)
logistic_regression_score = round(logistic_regression.score(test_x,test_y) * 100,2)

# Xg Boost
xg_boost = xgb.XGBClassifier(eta = clf_xgb.best_params_['eta'], gamma = clf_xgb.best_params_['gamma'], max_depth = clf_xgb.best_params_['max_depth'], min_child_weight=params['min_child_weight'])
xg_boost.fit(train_x,train_y)
prediction_xgb = xg_boost.predict(test_x)
xg_boost_score = round(xg_boost.score(test_x,test_y) * 100,2)

print('Decision Tree Performance: ', decision_tree_score)
print('Random Forest Performance: ', random_forest_score)
print('Logistic Regression Performance: ', logistic_regression_score)
print('XgBoost Performance: ', xg_boost_score)
```

Figure 14: Performs hyperparameter tuning using `GridSearchCV` for four machine learning models and evaluates their performance:

5 Results

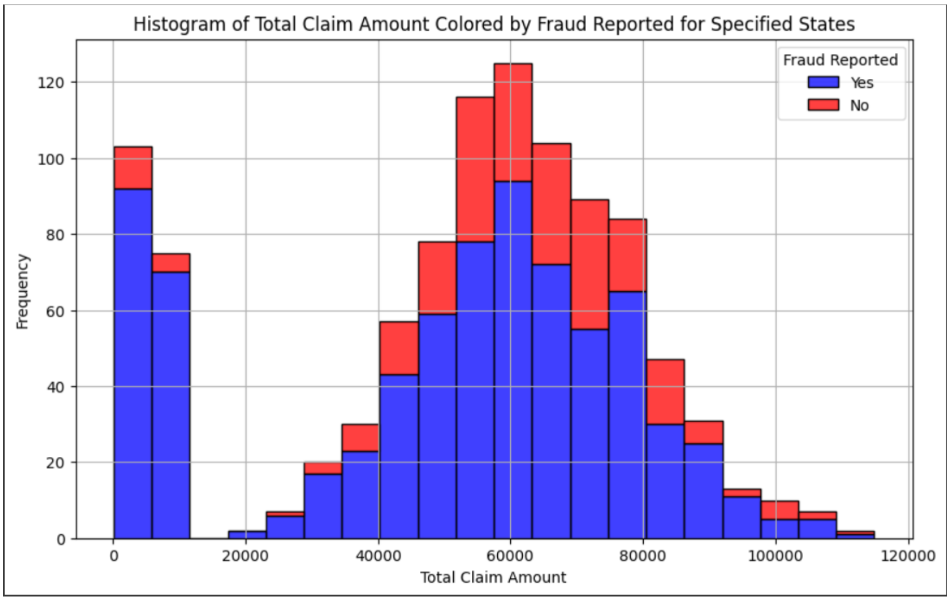

Figure 15 : Displaying the total claim amount**.**

```python
[ ] total_injury_claim_sum = filtered_df['total_claim_amount'].sum()

print(total_injury_claim_sum)

37867320
```

Figure 16: The total amount of claims after matching the state and year.