National College of Ireland

Vehicle Insurance Claim frequency and Amount Prediction through Machine Learning and Vehicle Analytics

MSc Research Project
Data Analytics

Arun Gangaramrao
Student ID: x22169202

School of Computing
National College of Ireland

Supervisor: Arghir Nicolae Moldovan

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Arun Gangaramrao
Student ID:	x22169202
Programme:	Data Analytics
Year:	2023
Module:	MSc Research Project
Supervisor:	Arghir Nicolae Moldovan
Submission Due Date:	14/12/2023
Project Title:	Vehicle Insurance Claim frequency and Amount Prediction through Machine Learning and Vehicle Analytics
Word Count:	1400
Page Count:	8

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:	Arun Gangaramrao
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):	

Vehicle Insurance Claim frequency and Amount Prediction through Machine Learning and Vehicle Analytics

Arun Gangaramrao
x22169202

1 Introduction

The comprehensive configuration manual serves as a detailed guide for conducting a research study, meticulously outlining the step-by-step process for system or device setup. It not only articulates the minimal setup essential for project success but also elucidates the necessary applications and packages required. With a focus on precision, the manual aims to provide users with a clear understanding of the machine configuration prerequisites essential for building and running models. Through this thorough documentation, users are empowered to seamlessly navigate the intricacies of the research study, ensuring a robust foundation for successful project execution.

2 Project Files Detail

Jupiter notebook framework is utilized in this research for data preparation, exploration, modelling and evaluation.

2.1 Case study-1:

• Case-study-1-file-2(Jupyter Notebook): Fetch the data in .CSV file then Data preparation, exploration, modelling and evaluation are done in this file using jupyter notebook. Data is loaded into the Jupyter Notebook from a CSV file. The dataset is cleaned, transformed, and organized as part of data preparation. Different algorithms are applied for assessment, examining performance metrics to improve the model.

The same above processes is carried out in Case study 2, 3, 4 and 5.

3 System Specification

A system specification is an in-depth document that lists all of the technical parameters and prerequisites for a system. It provides a thorough guide for the creation, application, and analysis of the architecture and functionalities of the system by including details on its parts, functionality, design, and other technical aspects.

Figure 1 and Figure 2 shows the system configuration of my system used to run this project.

Processor	Intel(R) Core(TM) i7-7700HQ CPU @ 2.80GHz 2.81 GHz
Installed RAM	16.0 GB (15.9 GB usable)
Device ID	BF0B3A95-42A7-4A2B-8E4F-45266355C731
Product ID	00342-41371-93111-AAOEM
System type	64-bit operating system, x64-based processor
Pen and touch	No pen or touch input is available for this display

Figure 1: Device Specification

Windows specifications

Edition	Windows 10 Home Single Language
Version	22H2
Installed on	26-10-2020
OS build	19045.3693
Experience	Windows Feature Experience Pack 1000.19053.1000.0

Figure 2: Windows Specification

Having a computer system with an Intel Core i5 processor or higher is recommended for optimal performance.It would be sufficient to have 8 GB of RAM for the most basic arrangement. Research tools and libraries are versatile, compatible with major operating systems such as Windows, macOS, and Linux, providing flexibility. Python must be installed together with necessary libraries like NumPy, Pandas, and Scikit-learn, as well as Jupyter Notebook. Although it would take a bit longer to process, the specification above would be sufficient to produce the intended outcomes. Still, the recommended setup guarantees error-free code execution.

4 Software Used

- Microsoft excel: Used for initial exploration.
- Jupyter Notebook[1]: For Exploration, exploratory data analysis, Processing, modelling and evaluation.

5 Download and Install

Installing Python is required initially, depending on the operating system; it is recommended to install the latest version [2]. The latest version of the file was downloaded and installed, which was Python 3.9 for Windows 10. After installing Python, a development environment is required in order to write, execute, and see the results of code. Probably

[1]Jupyter notebook: `https://jupyter.org`
[2]Python: `https://www.python.org/downloads/`

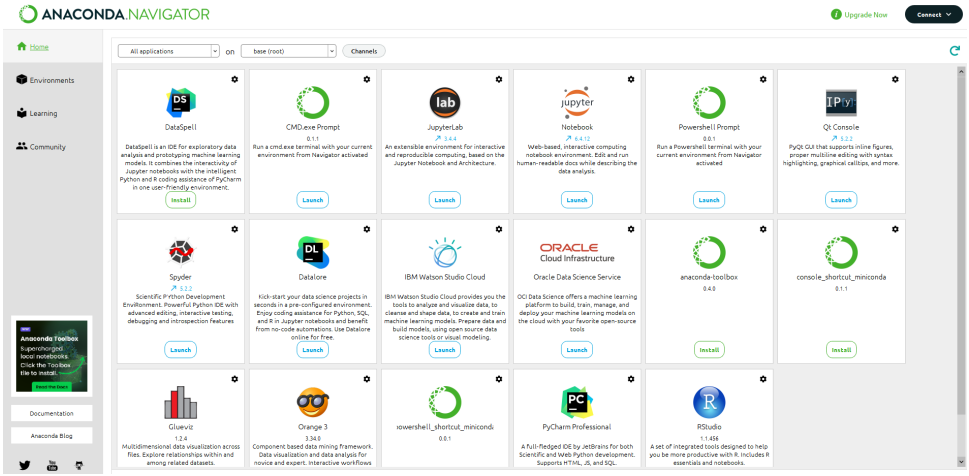

Figure 3: Anaconda Dashboard

the most popular and easiest to use platform is Jupyter Notebook. It comes pre-installed with the Anaconda [3] Python distribution, for which the appropriate installation can be downloaded based on the system. Figure 3 shows the Anaconda dashboard, which includes pre-installed programs such as the Jupyter notebook. Creating a new Python file in the Jupyter Notebook is the initial step in writing Python code.

6 Project Development

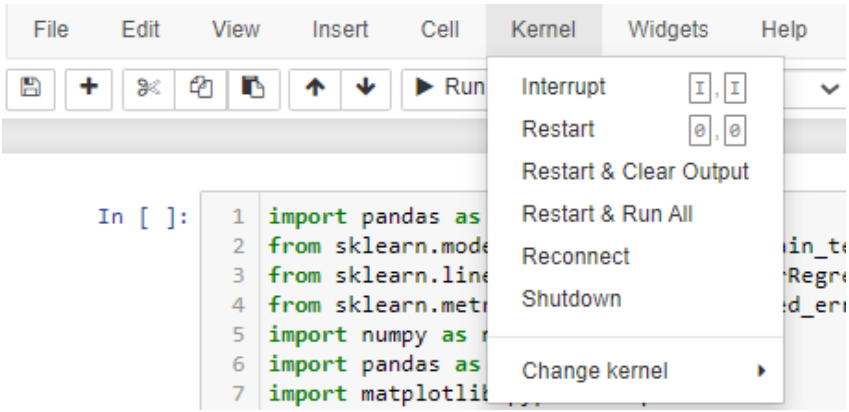

Figure 4: Jupyter Notebook Kernel run options

Once all of these steps are finished, you may open the Jupyter notebook. Click the new button at the top of the file to load the programmed file, and use the file reference that was supplied in the code section to load the file. You can choose to run each cell separately or all of them at once. In order to install a package, use the command "pip install package-name." Figure 4 shows the option to run the jupyter notebook file.

[3] Anaconda: `https://problemsolvingwithpython.com/01-Orientation/01.03-Installing-Anaconda-on-Windows/`

6.1 Importing Library

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import plotly.graph_objects as go
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import precision_score, recall_score, f1_score
from sklearn.metrics import confusion_matrix
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestClassifier, RandomForestRegressor
from sklearn.ensemble import AdaBoostClassifier
from imblearn.over_sampling import SMOTE
from sklearn.metrics import roc_curve, auc

```

Figure 5: Packages used in Classification

```
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import plotly.graph_objects as go
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestRegressor
from sklearn.svm import SVR
from sklearn.tree import DecisionTreeRegressor
from sklearn.linear_model import BayesianRidge
from sklearn.preprocessing import OneHotEncoder

```

Figure 6: Packages used in Regression

Using new functions in Jupyter Notebook requires importing libraries. The library name is used after the import keyword. Typical libraries are Matplotlib for data visualization, Pandas for data processing, and NumPy for numerical computations. This improves Jupyter's functionality and makes it a flexible tool for a range of data science applications.

Figure 5 shows the imports carried out for classification task and Figure 6 shows the import carried for regression task

```
1
2 data1 = pd.read_csv(r'train.csv',encoding='latin-1')
3 data1
```

Figure 7: Procedure to Fetch the File in Jupyter Notebook

6.2 Importing Files

Figure 7 demonstrates how to get the excel file in the notebook. The pd.readcsv() function is employed, the path should be set to the local directory where the file is placed. Similar code was carried out for other case studies also.

6.3 Processing

Data Inspection Upon initial examination, the dataset information is obtained using the .info() method, revealing three distinct data types: float64, int64, and object. Further insights are extracted using .describe(). To facilitate model training, categorical values undergo encoding via the Label Encoder.

Handling Missing Values Identification of missing and null values is conducted through .isna(), .isnull(), and .sum() functions, enabling a comprehensive overview of the dataset's data integrity.

Correlation Analysis and Feature Selection A correlation analysis is generated to examine relationships between target and independent variables. Employing a variance threshold of 0.1, essential features are selected for modeling, optimizing the dataset for predictive analytics.

Classification Dataset Refinement A targeted technique is utilized for binary classification in the setting of a classification dataset that has several unique values in the target variable. The dataset is streamlined for binary classification tasks by carefully removing target variable values greater than 1.

Normalization Using Standard Scalar The Standard Scaler is used to provide consistent feature scaling during normalization, a crucial preprocessing step that improves model performance and stability during training.

Regression Model Evaluation Metrics Customized error metric functions are introduced for regression tasks in order to evaluate model performance. These include the Root Relative Squared Error (RRSE) and Relative Absolute Error (RAE), which offer more detailed information about the precision and dependability of the regression model.

The packages or libraries to perform the above tasks are shown in the Figure 5 and Figure 6

6.4 Modeling

Prior to modeling, the target variable is put into the x-data and y-data functions together with the relevant predictive variables that were extracted from the recursive feature elimination. The test train is divided and fed into the models after smote is used to balance the data if there is an imbalance.

```
In [ ]:
print("-----------Information-----------")
print(data1.info())

In [ ]:
data1.describe()

In [ ]:
print("----------Missing value-----------")
print(data1.isna().sum())
print("----------Null value-----------")
print(data1.isnull().sum())

In [ ]:
numerical = [var for var in data1.columns if data1[var].dtype!='O']
categorical = [var for var in data1.columns if data1[var].dtype == 'O']
print("Numerical: ", numerical)
print("Caregorical: ", categorical)

In [ ]:

columns_to_encode = ['Insured.sex', 'Marital', 'Car.use', 'Region']

le = LabelEncoder()
for col in columns_to_encode:
    le.fit(data1[col])
    data1[f'{col}_encoded'] = le.transform(data1[col])
    print(f'{col} unique',data1[col].unique())

data1.drop(columns=['Insured.sex', 'Marital', 'Car.use', 'Region'], inplace=True)
data1.head()

In [ ]:

threshold_value = 0.1
selector = VarianceThreshold(threshold=threshold_value)

X_high_variance = selector.fit_transform(data1)

selected_feature_indices = selector.get_support(indices=True)


selected_feature_names = pd.DataFrame(data1, columns=feature_names).columns[selected_feature_indices]

print("Selected Feature Names:")
print(selected_feature_names)

In [ ]:
data1 = data1.drop('AMT_Claim', axis=1)

In [ ]:
data2 = data1[(data1['NB_Claim'] != 2) & (data1['NB_Claim'] != 3)]
```

Figure 8: Code Snippet of Early Stage of Classification Modeling

This project involves both regression and classification tasks for 3 datasets. For classification, logistic regression, random forest, and Adaboost models are utilized. In regression, decision tree, random forest regressor, gradient boost regressor, SVR, and Bayesian regressor models are employed. Each algorithm incorporates specific techniques, with the procedures iterated for other algorithms in the analysis.

Figure 8 and Figure 9 shows the early stages of classification and regression tasks.

6.4.1 Classification Models

Here, for dataset 1 and dataset 2 both multiclass and binary classification are carried out, for dataset 3 only binary classifcation is carried out. The processed dataset is used to train the selected models. A variety of performance metrics are used to assess the models after the training phase, including as accuracy, AUC value and F1 Score. Receiver Operating Characteristic (ROC) curves and confusion matrices are also produced.

Figure 10 shows the Random forest Classifier code snippet

```
In [ ]:
data = pd.merge(data1, data2, on='PolicyID', how='outer')

data
```

```
In [ ]:
numerical = [var for var in data.columns if data[var].dtype!='O']
categorical = [var for var in data.columns if data[var].dtype == 'O']
print("Numerical: ", numerical)
print("Caregorical: ", categorical)
```

```
In [ ]:
columns_to_encode = ['Power']
le = LabelEncoder()
for col in columns_to_encode:
    le.fit(data1[col])
    data[f'{col}_encoded'] = le.transform(data[col])
    print(f'{col} unique',data[col].unique())

data.drop(columns=['Power'], inplace=True)
data.describe()
```

```
In [ ]:
encoder = OneHotEncoder()
one_hot_encoded = encoder.fit_transform(data[['Brand','Gas','Region']])
one_hot_df = pd.DataFrame(one_hot_encoded.toarray(), columns=encoder.get_feature_names(['Brand','Gas','Region']))
data = pd.concat([data, one_hot_df], axis=1)

print(data)

```

```
In [ ]:
correlation_matrix = data.corr()
plt.figure(figsize=(12, 10))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', linewidths=0.5)
plt.title('Correlation Matrix Heatmap')
plt.show()

```

```
In [ ]:
print("----------Missing value----------")
print(data.isna().sum())
print("----------Null value----------")
print(data.isnull().sum())
```

```
In [ ]:
data['ClaimAmount'].fillna(0, inplace=True)

print("----------Missing value----------")
print(data.isna().sum())
print("----------Null value----------")
print(data.isnull().sum())
```

Figure 9: Code Snippet of Early Stage of Regression Modeling

```
# Random Forest Classifier
RClassifier = RandomForestClassifier(n_estimators=100, random_state=42)
RClassifier.fit(X_train, y_train)
predictions = RClassifier.predict(X_test)
accuracy = accuracy_score(y_test, predictions)
print("Accuracy: {:.4f}".format(accuracy))
print("Classification Report:\n", classification_report(y_test, predictions))
precision = precision_score(y_test, predictions, average='weighted')
recall = recall_score(y_test, predictions, average='weighted')
f1 = f1_score(y_test, predictions, average='weighted')
print("Precision per class:", precision)
print("Recall per class:", recall)
print("F1 Score per class:", f1)
```

Figure 10: Random Forest Classifier Model

6.4.2 Regression Models

Here dataset 1 and dataset 2 are considered for regression. GridSearchCV is employed for hyperparameter tuning, optimizing the models for better predictive performance. The evaluation metrics for each model, including Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), R-squared (R2) Score, Relative Absolute Error (RAE), and Root Relative Squared Error (RRSE), are calculated and printed.

Figure 11 shows the Decision Tree Regressor code snippet

```python
#Decision Tree Regressor
regressor = DecisionTreeRegressor(random_state=42)
param_grid = {
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
}

grid_search = GridSearchCV(regressor, param_grid, cv=5, scoring='neg_mean_squared_error')

grid_search.fit(X_train_scaled, y_train)
best_params = grid_search.best_params_

best_regressor = grid_search.best_estimator_

predictions = best_regressor.predict(X_test_scaled)

mae = mean_absolute_error(y_test, predictions)
mse = mean_squared_error(y_test, predictions)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, predictions)
rae = relative_absolute_error(y_test, predictions)
rrse = root_relative_squared_error(y_test, predictions)

print("Best Hyperparameters:", best_params)

print("Relative Absolute Error (RAE):", rae)
print("Root Relative Squared Error (RRSE):", rrse)
print(f"Mean Absolute Error (MAE): {mae:.2f}")
print(f"Mean Squared Error (MSE): {mse:.2f}")
print(f"Root Mean Squared Error (RMSE): {rmse:.2f}")
print(f"R-squared (R2) Score: {r2:.2f}")

```

Figure 11: Decision Tree Regressor Model