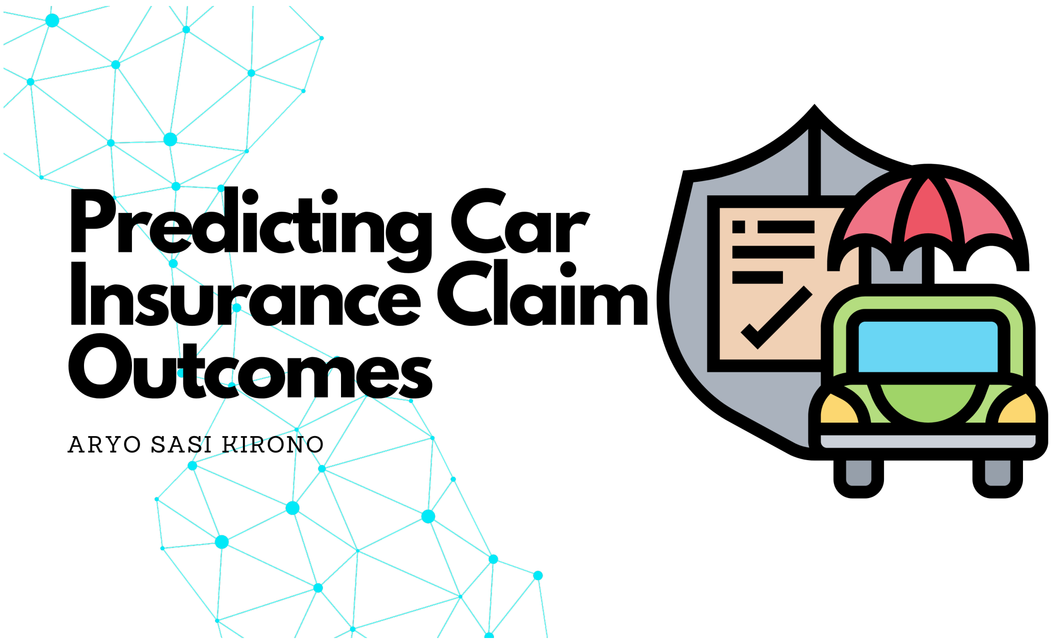
PROJECT 1

April 23, 2024



1 Business Understanding

Insurance companies invest a lot of time and money into optimizing their pricing and accurately estimating the likelihood that customers will make a claim. In many countries insurance it is a legal requirement to have car insurance in order to drive a vehicle on public roads, so the market is very large!

Knowing all of this, On the Road car insurance have requested your services in building a model to predict whether a customer will make a claim on their insurance during the policy period. As they have very little expertise and infrastructure for deploying and monitoring machine learning models, they've asked you to identify the best performing model, as measured by accuracy, so they can start with the model in production.

They have supplied you with their customer data as a csv file called `car_insurance.csv`, along with a table detailing the column names and descriptions below.

1.1 The dataset

Column	Description
id	Unique client identifier
age	Client's age:
gender	Client's gender:
driving_experience	Years the client has been driving:
education	Client's level of education:
income	Client's income level:
credit_score	Client's credit score (between zero and one)
vehicle_ownership	Client's vehicle ownership status:
vehcile_year	Year of vehicle registration:
married	Client's marital status:
children	Client's number of children
postal_code	Client's postal code
annual_mileage	Number of miles driven by the client each year
vehicle_type	Type of car:
speeding_violations	Total number of speeding violations received by the client
duis	Number of times the client has been caught driving under the influence of alcohol
past_accidents	Total number of previous accidents the client has been involved in
outcome	Whether the client made a claim on their car insurance (response variable):

[105]:
```
# Modules to handle table-like data and matrices
import pandas as pd
import numpy as np

# Modelling Algorithms Modules
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression

# Modelling Helpers Modules
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import LabelEncoder


# Visualization Modules
import seaborn as sns
import matplotlib.pyplot as plt
```

[106]:
```
cars = pd.read_csv('car_insurance.csv')
cars.head()
```

[106]:
```
       id  age  gender driving_experience    education        income  \
0  569520    3       0               0-9y  high school   upper class
```

```
1  750365    0       1             0-9y         none        poverty
2  199901    0       0             0-9y  high school  working class
3  478866    0       1             0-9y   university  working class
4  731664    1       1           10-19y         none  working class

   credit_score  vehicle_ownership vehicle_year  married  children  \
0      0.629027                1.0   after 2015      0.0       1.0
1      0.357757                0.0  before 2015      0.0       0.0
2      0.493146                1.0  before 2015      0.0       0.0
3      0.206013                1.0  before 2015      0.0       1.0
4      0.388366                1.0  before 2015      0.0       0.0

   postal_code  annual_mileage vehicle_type  speeding_violations  duis  \
0        10238         12000.0        sedan                    0     0
1        10238         16000.0        sedan                    0     0
2        10238         11000.0        sedan                    0     0
3        32765         11000.0        sedan                    0     0
4        32765         12000.0        sedan                    2     0

   past_accidents  outcome
0               0      0.0
1               0      1.0
2               0      0.0
3               0      0.0
4               1      1.0
```

[107]:
```
cars.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 18 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   id                   10000 non-null  int64
 1   age                  10000 non-null  int64
 2   gender               10000 non-null  int64
 3   driving_experience   10000 non-null  object
 4   education            10000 non-null  object
 5   income               10000 non-null  object
 6   credit_score         9018 non-null   float64
 7   vehicle_ownership    10000 non-null  float64
 8   vehicle_year         10000 non-null  object
 9   married              10000 non-null  float64
 10  children             10000 non-null  float64
 11  postal_code          10000 non-null  int64
 12  annual_mileage       9043 non-null   float64
 13  vehicle_type         10000 non-null  object
 14  speeding_violations  10000 non-null  int64
```

```
 15   duis                    10000 non-null   int64
 16   past_accidents          10000 non-null   int64
 17   outcome                 10000 non-null   float64
dtypes: float64(6), int64(7), object(5)
memory usage: 1.4+ MB
```

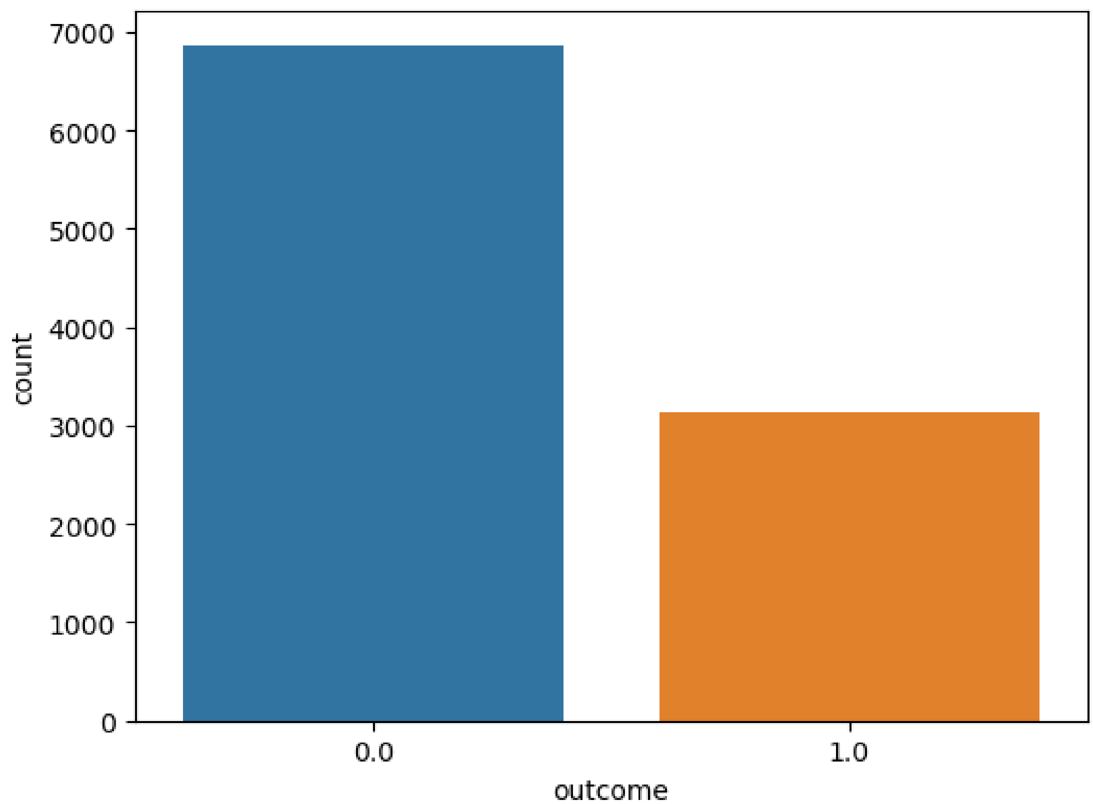
2 Data Understanding

Check the balance between outcome

1 : claims made, 0 : no claims

[108]:
```
sns.countplot(x='outcome', data=cars)
```

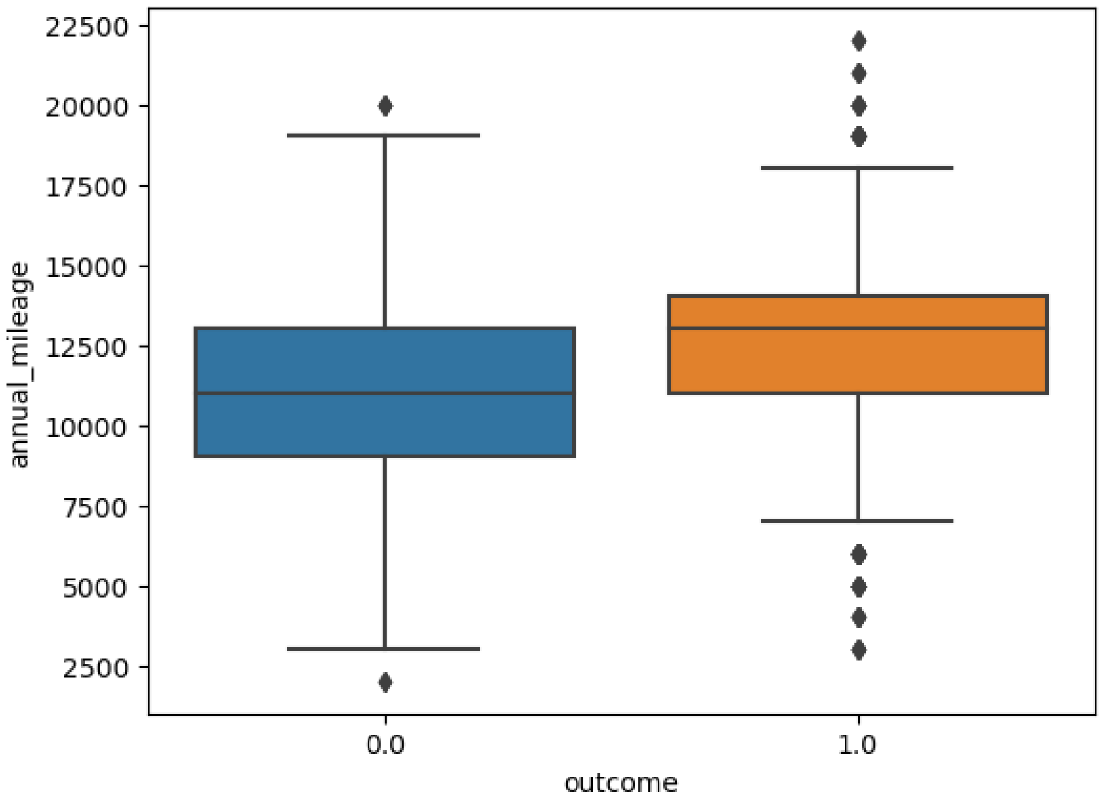
[108]:
```
<Axes: xlabel='outcome', ylabel='count'>
```



Check the presence and distribution of outliers using boxplot

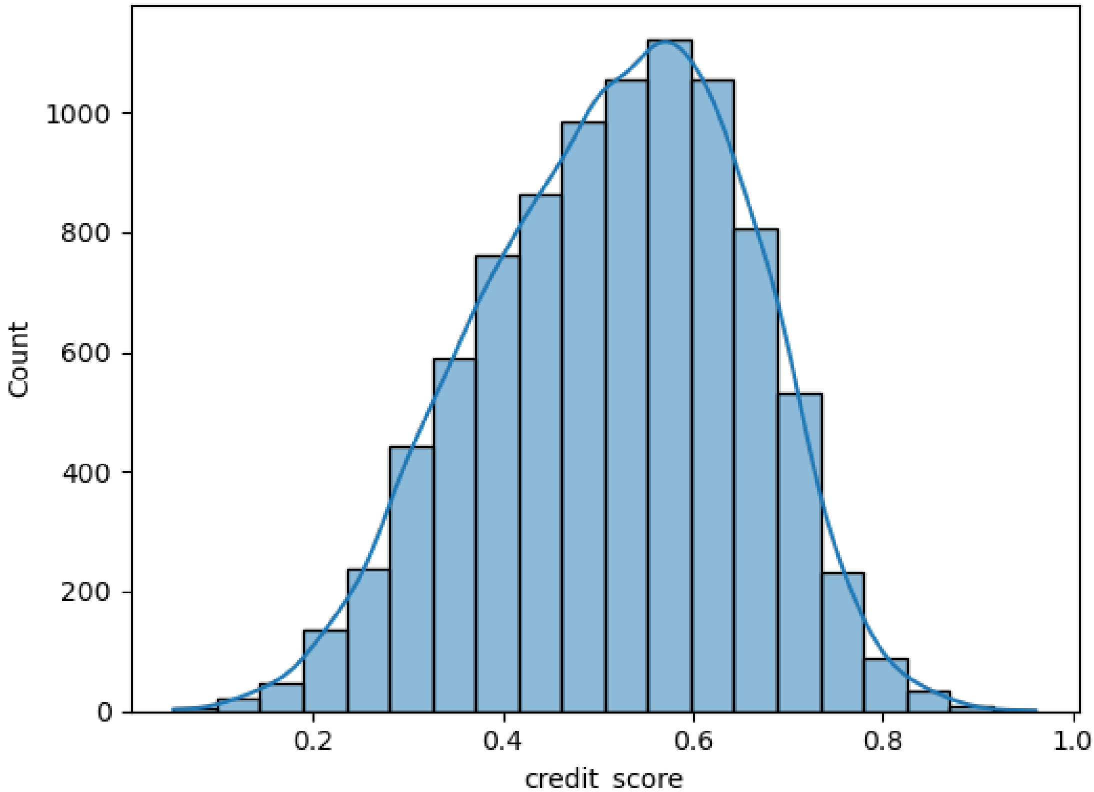
[109]:
```
sns.boxplot(x='outcome', y='annual_mileage', data=cars)
```

[109]:
```
<Axes: xlabel='outcome', ylabel='annual_mileage'>
```



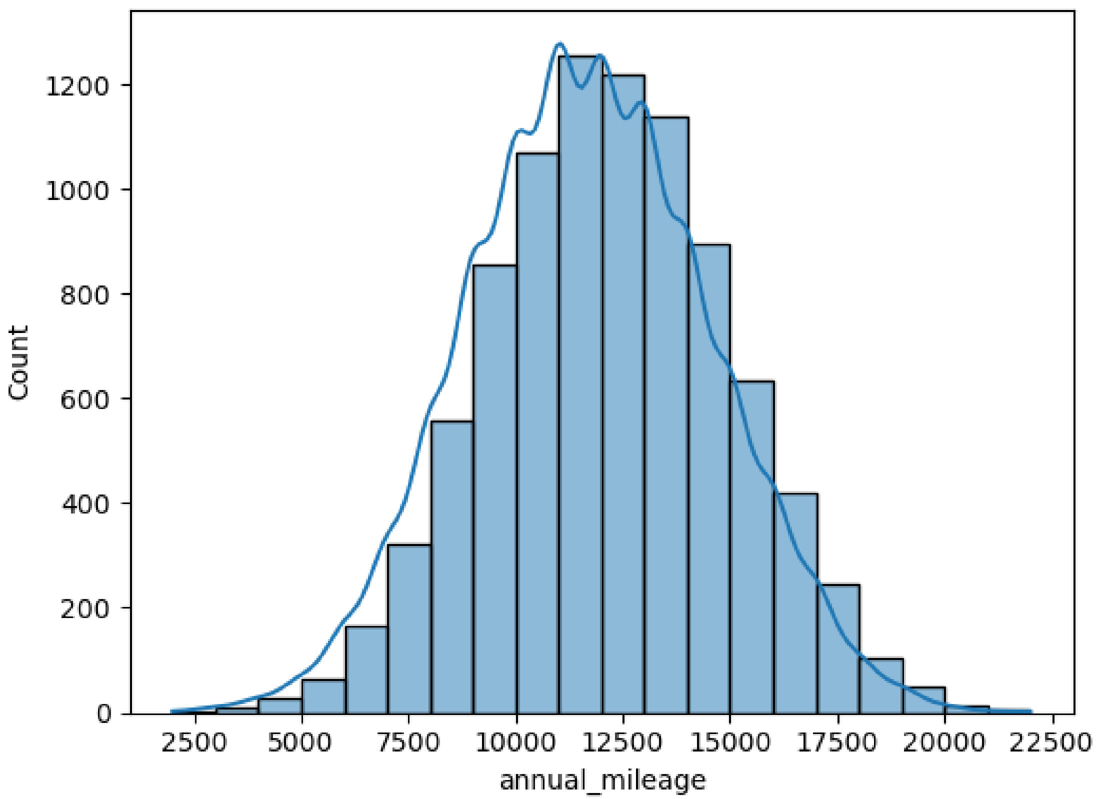
[110]:
```
sns.histplot(cars['credit_score'], bins=20, kde=True)
```

[110]: <Axes: xlabel='credit_score', ylabel='Count'>



[111]:
```
sns.histplot(cars['annual_mileage'], bins=20, kde=True)
```

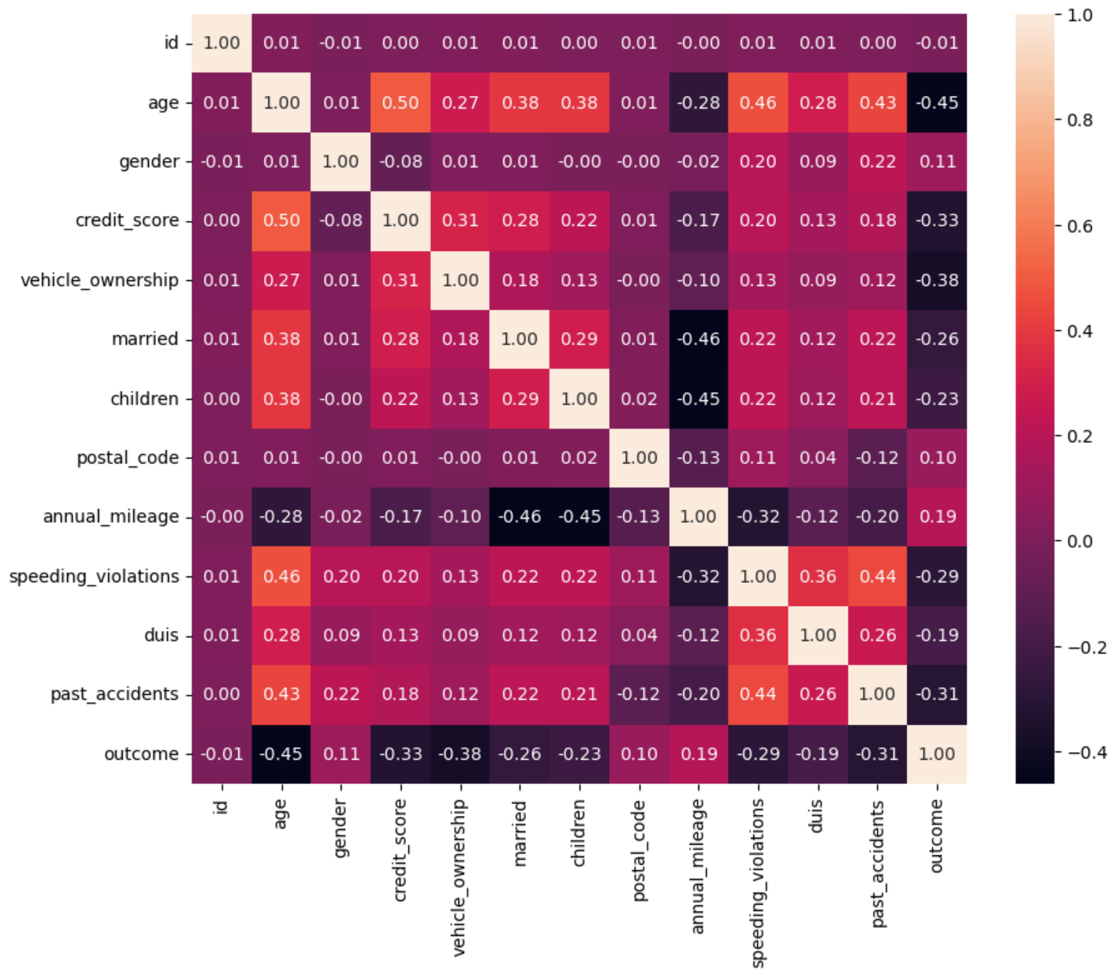
[111]: <Axes: xlabel='annual_mileage', ylabel='Count'>



[112]:
```
correlation = cars.corr()
plt.figure(figsize=(10,8))
sns.heatmap(correlation, annot=True, fmt='.2f')
```

```
C:\Users\Aryo Sasi\AppData\Local\Temp\ipykernel_2436\2857633541.py:1:
FutureWarning: The default value of numeric_only in DataFrame.corr is
deprecated. In a future version, it will default to False. Select only valid
columns or specify the value of numeric_only to silence this warning.
  correlation = cars.corr()
```

[112]: <Axes: >



3 Data Preparation

handle missing value with mean function

[113]:
```
cars['credit_score'].fillna(cars['credit_score'].mean(), inplace = True)
cars['annual_mileage'].fillna(cars['annual_mileage'].mean(), inplace = True)
cars.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 18 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   id                   10000 non-null  int64
 1   age                  10000 non-null  int64
 2   gender               10000 non-null  int64
 3   driving_experience   10000 non-null  object
```

```
 4   education           10000 non-null  object
 5   income              10000 non-null  object
 6   credit_score        10000 non-null  float64
 7   vehicle_ownership   10000 non-null  float64
 8   vehicle_year        10000 non-null  object
 9   married             10000 non-null  float64
 10  children            10000 non-null  float64
 11  postal_code         10000 non-null  int64
 12  annual_mileage      10000 non-null  float64
 13  vehicle_type        10000 non-null  object
 14  speeding_violations 10000 non-null  int64
 15  duis                10000 non-null  int64
 16  past_accidents      10000 non-null  int64
 17  outcome             10000 non-null  float64
dtypes: float64(6), int64(7), object(5)
memory usage: 1.4+ MB
```

Handle outliers in the annual_mileage column using RobustScaler

[114]:
```python
from sklearn.preprocessing import RobustScaler

scaler = RobustScaler()
cars['annual_mileage_scaled'] = scaler.fit_transform(cars[['annual_mileage']])
cars.head()
```

[114]:
```
       id  age  gender driving_experience    education         income  \
0  569520    3       0               0-9y  high school    upper class
1  750365    0       1               0-9y         none        poverty
2  199901    0       0               0-9y  high school  working class
3  478866    0       1               0-9y   university  working class
4  731664    1       1             10-19y         none  working class

   credit_score  vehicle_ownership vehicle_year  married  children  \
0      0.629027                1.0   after 2015      0.0       1.0
1      0.357757                0.0  before 2015      0.0       0.0
2      0.493146                1.0  before 2015      0.0       0.0
3      0.206013                1.0  before 2015      0.0       1.0
4      0.388366                1.0  before 2015      0.0       0.0

   postal_code  annual_mileage vehicle_type  speeding_violations  duis  \
0        10238         12000.0        sedan                    0     0
1        10238         16000.0        sedan                    0     0
2        10238         11000.0        sedan                    0     0
3        32765         11000.0        sedan                    0     0
4        32765         12000.0        sedan                    2     0

   past_accidents  outcome  annual_mileage_scaled
0               0      0.0               0.100999
```

```
1                0      1.0                 1.434332
2                0      0.0                -0.232334
3                0      0.0                -0.232334
4                1      1.0                 0.100999
```

Create a funvtion to encode categorical columns using LabelEncoder

[115]:
```
def columns_le(data, columns):
    le = LabelEncoder()
    for column in columns:
        data[f'{column}_encoded'] = le.fit_transform(data[column]) # encode the␣
↪column
        data.drop(column, axis = 1, inplace=True) # drop the previous column
    return data
```

[116]:
```
columns_to_encode = ['driving_experience', 'education', 'income',␣
↪'vehicle_year', 'vehicle_type']
encoded_cars = columns_le(cars, columns_to_encode)
encoded_cars.drop('annual_mileage', axis=1, inplace=True)
encoded_cars.head()
```

[116]:
```
       id  age  gender  credit_score  vehicle_ownership  married  children  \
0  569520    3       0      0.629027                1.0      0.0       1.0
1  750365    0       1      0.357757                0.0      0.0       0.0
2  199901    0       0      0.493146                1.0      0.0       0.0
3  478866    0       1      0.206013                1.0      0.0       1.0
4  731664    1       1      0.388366                1.0      0.0       0.0

   postal_code  speeding_violations  duis  past_accidents  outcome  \
0        10238                    0     0               0      0.0
1        10238                    0     0               0      1.0
2        10238                    0     0               0      0.0
3        32765                    0     0               0      0.0
4        32765                    2     0               1      1.0

   annual_mileage_scaled  driving_experience_encoded  education_encoded  \
0               0.100999                           0                  0
1               1.434332                           0                  1
2              -0.232334                           0                  0
3              -0.232334                           0                  2
4               0.100999                           1                  1

   income_encoded  vehicle_year_encoded  vehicle_type_encoded
0               2                     0                     0
1               1                     1                     0
2               3                     1                     0
3               3                     1                     0
```

```
4                3                      1                      0
```

Column After Label Encoder	Description
driving_experience	Years the client has been driving:
education	Client's level of education:
income	Client's income level:
vehcile_year	Year of vehicle registration:
vehicle_type	Type of car:

[117]:

```
encoded_cars.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 18 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   id                          10000 non-null  int64
 1   age                         10000 non-null  int64
 2   gender                      10000 non-null  int64
 3   credit_score                10000 non-null  float64
 4   vehicle_ownership           10000 non-null  float64
 5   married                     10000 non-null  float64
 6   children                    10000 non-null  float64
 7   postal_code                 10000 non-null  int64
 8   speeding_violations         10000 non-null  int64
 9   duis                        10000 non-null  int64
 10  past_accidents              10000 non-null  int64
 11  outcome                     10000 non-null  float64
 12  annual_mileage_scaled       10000 non-null  float64
 13  driving_experience_encoded  10000 non-null  int32
 14  education_encoded           10000 non-null  int32
 15  income_encoded              10000 non-null  int32
 16  vehicle_year_encoded        10000 non-null  int32
 17  vehicle_type_encoded        10000 non-null  int32
dtypes: float64(6), int32(5), int64(7)
memory usage: 1.2 MB
```

4 Modelling

4.1 Using Support Vector Machine, Random Forest, Logistic Regression

[118]:

```
X = encoded_cars.drop(['outcome', 'id'], axis=1)
y = encoded_cars['outcome']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪stratify=y, random_state=9)
```

Performing Hyperparameter tuning using GridSearch to find the best paramaters and model.

[119]:
```
model_params = {
    'svc': {
        'model': SVC(gamma='auto'),
        'params' : {
            'C': [1,10,20]
        }
    },
    'random_forest': {
        'model': RandomForestClassifier(),
        'params' : {
            'n_estimators': [1,5,10]
        }
    },
    'logistic_regression' : {
        'model': LogisticRegression(solver='liblinear',multi_class='auto'),
        'params': {
            'C': [1,5,10]
        }
    }
}
```

5 Evaluation

[127]:
```
scores = []

for model_name, mp in model_params.items():
    clf = GridSearchCV(mp['model'], mp['params'], cv = 5,␣
 ↪return_train_score=False)
    clf.fit(X_train, y_train)
    scores.append({
        'model':model_name,
        'best_score':clf.best_score_,
        'best_params':clf.best_params_
    })
```

[129]:
```
df = pd.DataFrame(scores, columns=['model', 'best_score', 'best_params'])
df
```

[129]:
```
                 model  best_score           best_params  \
0                  svc    0.848000            {'C': 1}
1        random_forest    0.823125  {'n_estimators': 10}
2  logistic_regression    0.820625            {'C': 1}

                            best_estimator  best_index
0                   SVC(C=1, gamma='auto')           0
```

```
1  (DecisionTreeClassifier(max_features='sqrt', r…          2
2        LogisticRegression(C=1, solver='liblinear')          0
```

Build a simple prediction system

using the first row index where id is 569520 and the outcome is 0 : no claims

[122]:
```python
encoded_cars.head()
```

[122]:
```
       id  age  gender  credit_score  vehicle_ownership  married  children  \
0  569520    3       0      0.629027                1.0      0.0       1.0
1  750365    0       1      0.357757                0.0      0.0       0.0
2  199901    0       0      0.493146                1.0      0.0       0.0
3  478866    0       1      0.206013                1.0      0.0       1.0
4  731664    1       1      0.388366                1.0      0.0       0.0

   postal_code  speeding_violations  duis  past_accidents  outcome  \
0        10238                    0     0               0      0.0
1        10238                    0     0               0      1.0
2        10238                    0     0               0      0.0
3        32765                    0     0               0      0.0
4        32765                    2     0               1      1.0

   annual_mileage_scaled  driving_experience_encoded  education_encoded  \
0               0.100999                           0                  0
1               1.434332                           0                  1
2              -0.232334                           0                  0
3              -0.232334                           0                  2
4               0.100999                           1                  1

   income_encoded  vehicle_year_encoded  vehicle_type_encoded
0               2                     0                     0
1               1                     1                     0
2               3                     1                     0
3               3                     1                     0
4               3                     1                     0
```

[123]:
```python
input_data = (3, 0, 0.629027, 1.0, 0.0, 1.0, 10238, 12000.0, 0, 0, 0, 0, 0, 2,
 0, 0)

svc = SVC(C=1, gamma='auto')

input_array = np.asarray(input_data)

input_reshaped = input_array.reshape(1,-1)

svc.fit(X_train, y_train)
```

```python
prediction = svc.predict(input_reshaped)
print(prediction)
```

```
[0.]
```

```
C:\Users\Aryo Sasi\anaconda3\Lib\site-packages\sklearn\base.py:464: UserWarning:
X does not have valid feature names, but SVC was fitted with feature names
  warnings.warn(
```

6 Conclusion

The purpose of this project is to build a model that predicts whether a customer will make an insurance claim. After testing several models, the best-performing one was the Support Vector Machine with a parameter C of 1, which achieved the highest accuracy of 84% in predicting customer insurance claim submissions. In testing simple predictions, the model also correctly predicted `No Claims` for the first row.


```
[ ]:
```