Performed By

Mohit T. Kumbhare
Pranay D. Sonkusare

Insurance Claim Status Prediction

17th January 2022

OVERVIEW

Insurance companies take risks over customers. Risk management is a very important aspect of the insurance industry. Insurers consider every quantifiable factor to develop profiles of high and low insurance risks. Insurers collect vast amounts of information about policyholders and analyse the data. As a Data scientist in an insurance company, you need to analyse the available data and predict whether to approve the insurance or not.

GOALS

1. Predict whether to approve the insurance or not.

Deep Learning

Deep learning is a subset of machine learning which is completely based on artificial neural networks.Deep Learning is a field of study which is focused on making machines mimic the human brain.In deep learning, we don't need to explicitly program everything. We will cover Deep Learning in-depth later.

Links

Dataset - data.csv

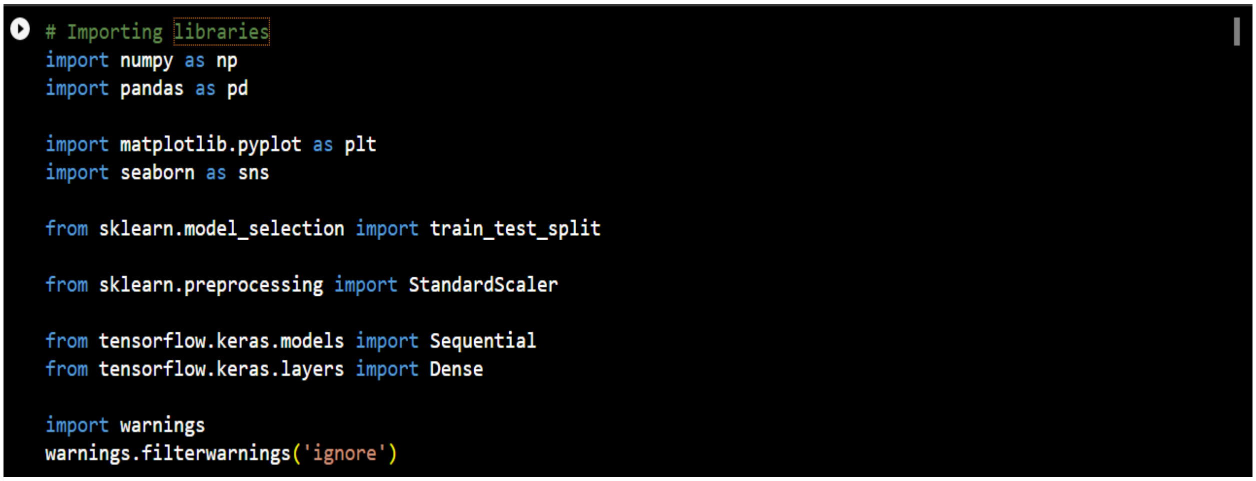
Performed Practical - Project.ipynb

Practical

1. About DataSet

Features	Description
ID	Unique identifier
Agency	Agency name
Agency Type	Type of travel insurance agency
Distribution Channel	Online/Offline distribution channel
Product Name	Travel insurance product name
Duration	Duration of travel
Destination	Destination of travel
Net sales	Net sales of travel insurance policies
Commision	The commission received by travel insurance agency
Gender	Traveller's gender
Age	Traveller's Age

2. Importing Libraries



```
# Importing libraries
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import StandardScaler

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

import warnings
warnings.filterwarnings('ignore')
```

a. NumPy - NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects.

b. Pandas - **Pandas** is an open source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language.

c. Matplotlib - Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python.

d. Seaborn - Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

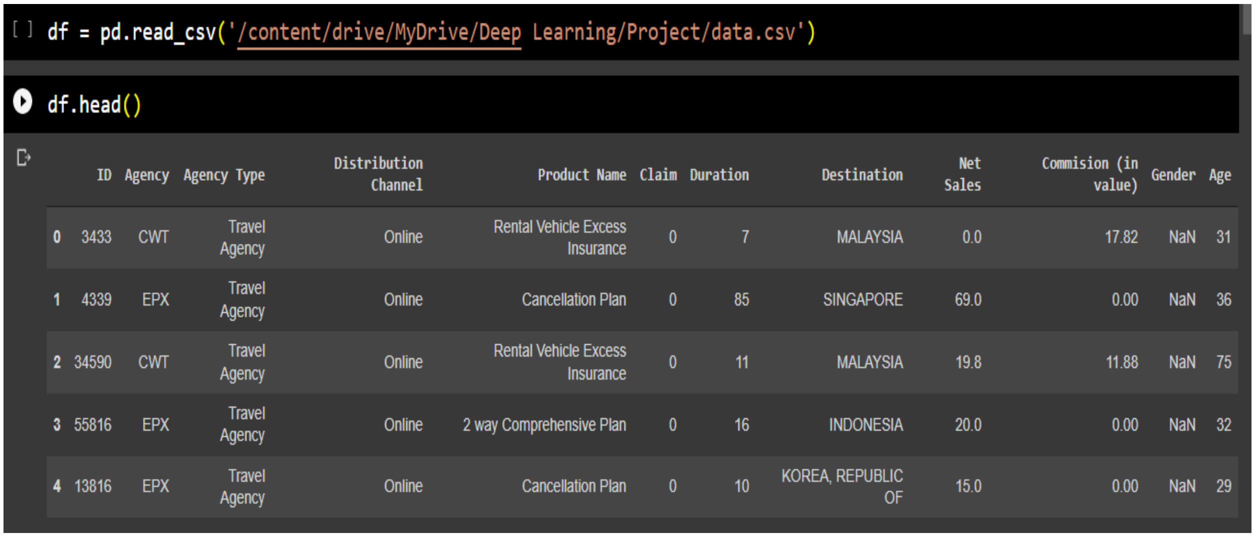
e. TrainTestSplit - Quick utility that wraps input validation and next(ShuffleSplit().split(X, y)) and application to input data into a single call for splitting (and optionally subsampling) data in a one liner.

f. Standard Scaler - Standardize features by removing the mean and scaling to unit variance.

g. Sequential - Sequential groups a linear stack of layers into a tf.keras.Model. Sequential provides training and inference features on this model.

h. Dense - Dense implements the operation: output = activation(dot(input, kernel) + bias) where activation is the element-wise activation function passed as the activation argument, kernel is a weights matrix created by the layer, and bias is a bias vector created by the layer (only applicable if use_bias is True).

3. Import Data And Analyse



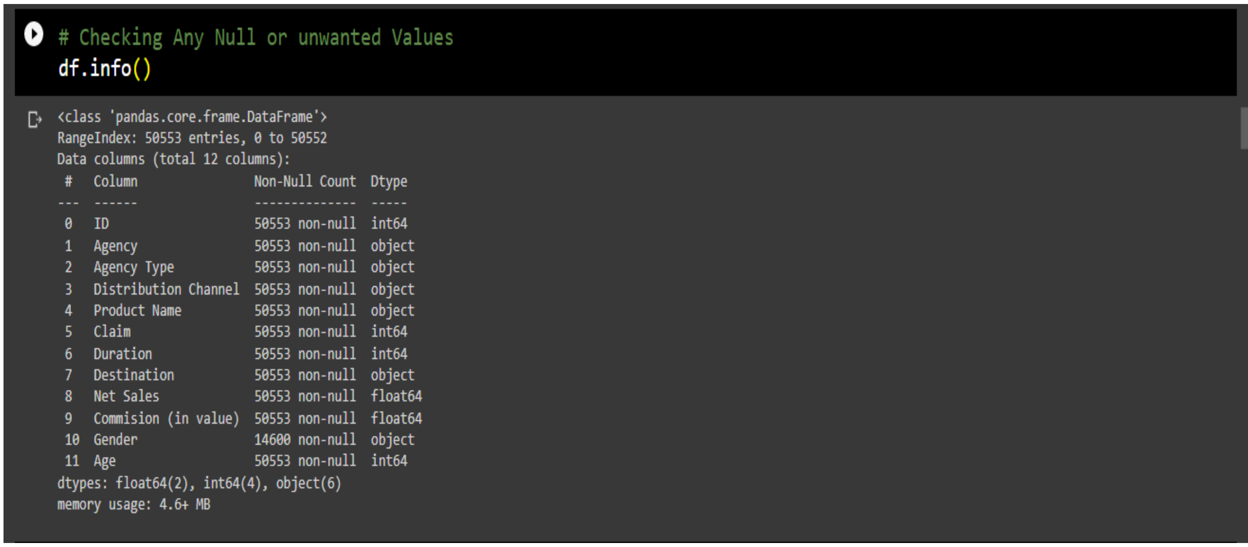
```
[ ] df = pd.read_csv('/content/drive/MyDrive/Deep Learning/Project/data.csv')
```

```
df.head()
```

	ID	Agency	Agency Type	Distribution Channel	Product Name	Claim	Duration	Destination	Net Sales	Commision (in value)	Gender	Age
0	3433	CWT	Travel Agency	Online	Rental Vehicle Excess Insurance	0	7	MALAYSIA	0.0	17.82	NaN	31
1	4339	EPX	Travel Agency	Online	Cancellation Plan	0	85	SINGAPORE	69.0	0.00	NaN	36
2	34590	CWT	Travel Agency	Online	Rental Vehicle Excess Insurance	0	11	MALAYSIA	19.8	11.88	NaN	75
3	55816	EPX	Travel Agency	Online	2 way Comprehensive Plan	0	16	INDONESIA	20.0	0.00	NaN	32
4	13816	EPX	Travel Agency	Online	Cancellation Plan	0	10	KOREA, REPUBLIC OF	15.0	0.00	NaN	29

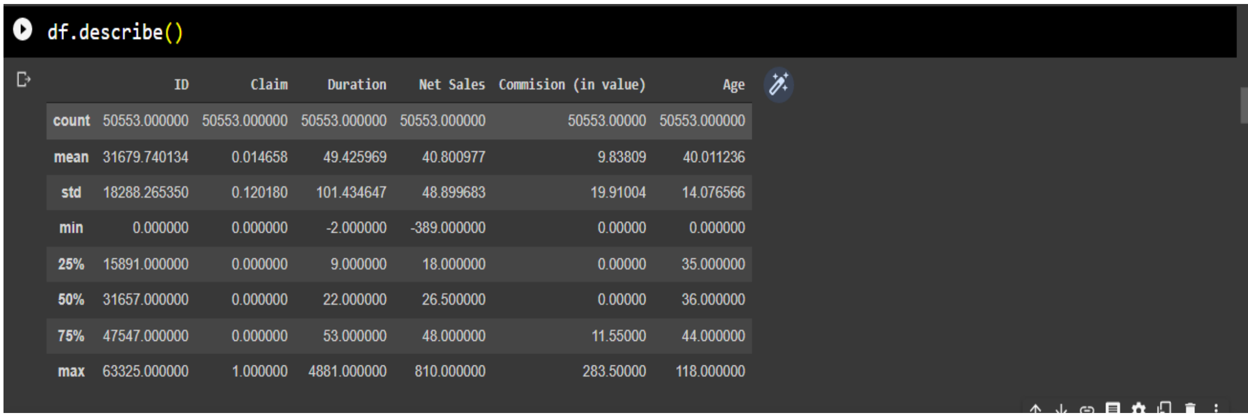
Here we import Data From drive using the pandas library and using df.head we see the first 5 data.

a. Check Null Values



```
# Checking Any Null or unwanted Values
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50553 entries, 0 to 50552
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   ID                    50553 non-null  int64
 1   Agency                50553 non-null  object
 2   Agency Type           50553 non-null  object
 3   Distribution Channel  50553 non-null  object
 4   Product Name          50553 non-null  object
 5   Claim                 50553 non-null  int64
 6   Duration              50553 non-null  int64
 7   Destination           50553 non-null  object
 8   Net Sales             50553 non-null  float64
 9   Commision (in value)  50553 non-null  float64
 10  Gender                14600 non-null  object
 11  Age                   50553 non-null  int64
dtypes: float64(2), int64(4), object(6)
memory usage: 4.6+ MB
```

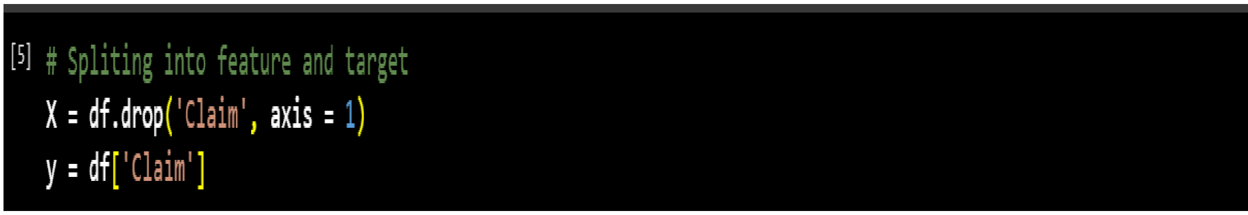


```
df.describe()
```

	ID	Claim	Duration	Net Sales	Commision (in value)	Age
count	50553.000000	50553.000000	50553.000000	50553.000000	50553.00000	50553.000000
mean	31679.740134	0.014658	49.425969	40.800977	9.83809	40.011236
std	18288.265350	0.120180	101.434647	48.899683	19.91004	14.076566
min	0.000000	0.000000	-2.000000	-389.000000	0.00000	0.000000
25%	15891.000000	0.000000	9.000000	18.000000	0.00000	35.000000
50%	31657.000000	0.000000	22.000000	26.500000	0.00000	36.000000
75%	47547.000000	0.000000	53.000000	48.000000	11.55000	44.000000
max	63325.000000	1.000000	4881.000000	810.000000	283.50000	118.000000

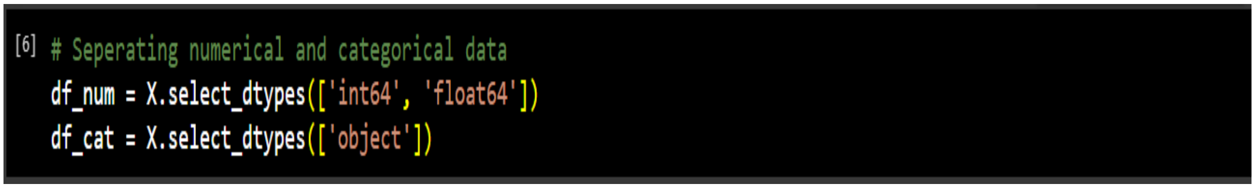
Using df.info we check any null values present in data and data type of data and using df.describe() check mean median of numerical data

4. Split Target and Feature



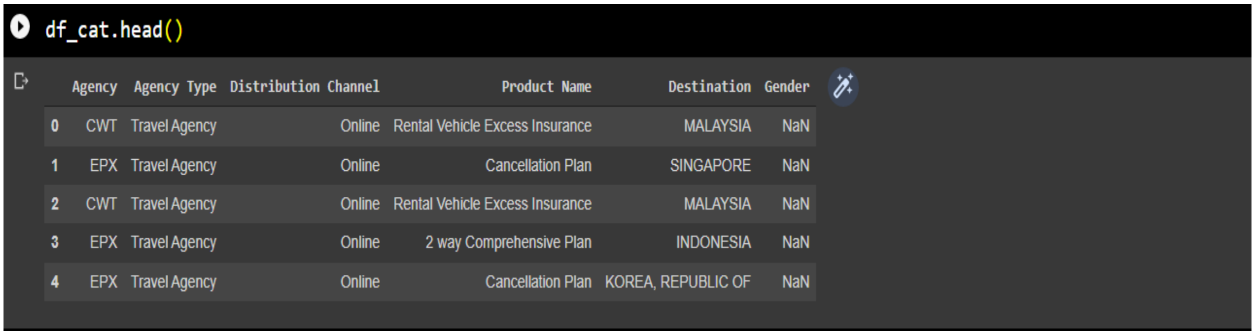
```
[5] # Spliting into feature and target
    X = df.drop('Claim', axis = 1)
    y = df['Claim']
```

5. Numerical And Categorical Data



```
[6] # Seperating numerical and categorical data
    df_num = X.select_dtypes(['int64', 'float64'])
    df_cat = X.select_dtypes(['object'])
```

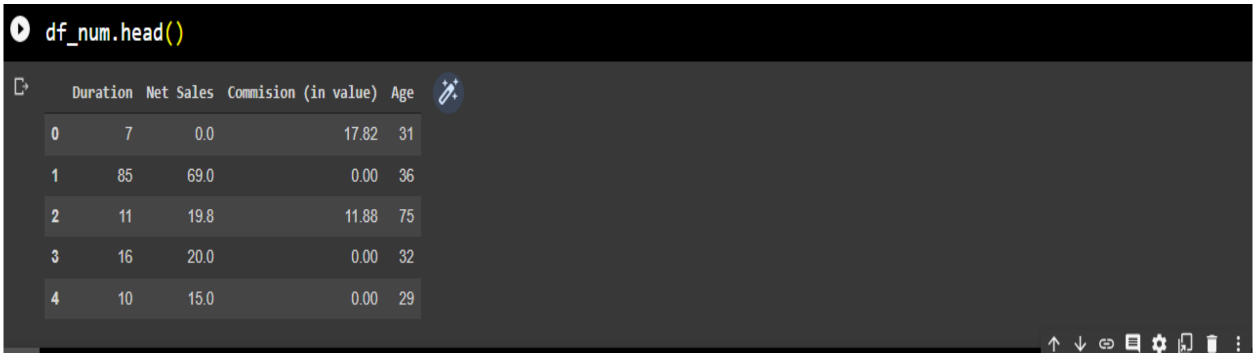
Categorical data -



```
df_cat.head()
```

	Agency	Agency Type	Distribution Channel	Product Name	Destination	Gender
0	CWT	Travel Agency	Online	Rental Vehicle Excess Insurance	MALAYSIA	NaN
1	EPX	Travel Agency	Online	Cancellation Plan	SINGAPORE	NaN
2	CWT	Travel Agency	Online	Rental Vehicle Excess Insurance	MALAYSIA	NaN
3	EPX	Travel Agency	Online	2 way Comprehensive Plan	INDONESIA	NaN
4	EPX	Travel Agency	Online	Cancellation Plan	KOREA, REPUBLIC OF	NaN

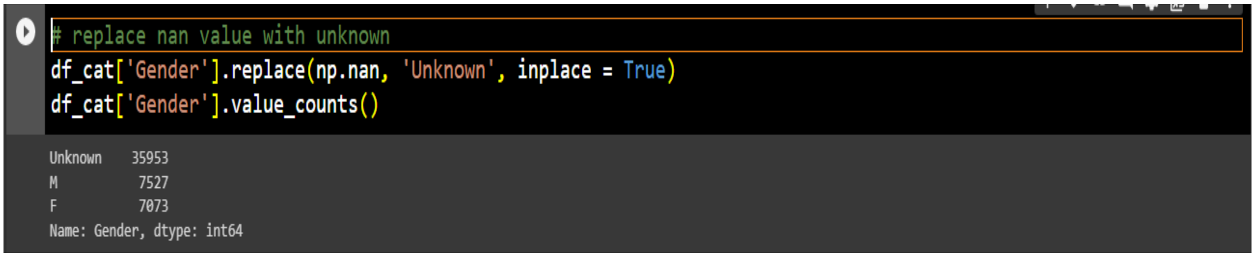
Numerical data -



```
df_num.head()
```

	Duration	Net Sales	Commision (in value)	Age
0	7	0.0	17.82	31
1	85	69.0	0.00	36
2	11	19.8	11.88	75
3	16	20.0	0.00	32
4	10	15.0	0.00	29

Now here in categorical data we see NaN values in Age so replace it with "Unknown"

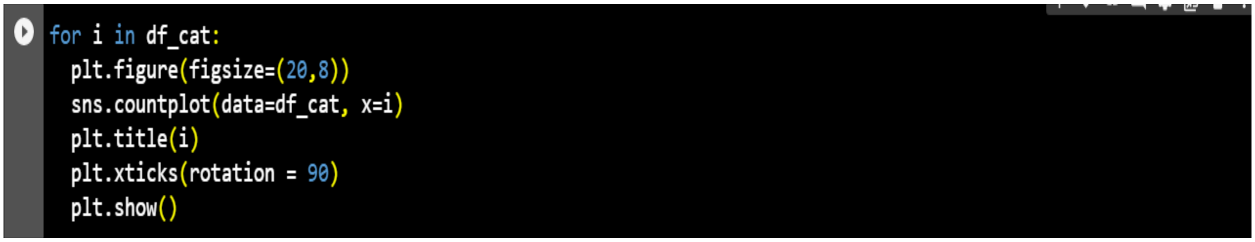


```
# replace nan value with unknown
df_cat['Gender'].replace(np.nan, 'Unknown', inplace = True)
df_cat['Gender'].value_counts()
```

```
Unknown    35953
M           7527
F           7073
Name: Gender, dtype: int64
```

6. Plot Categorical Data

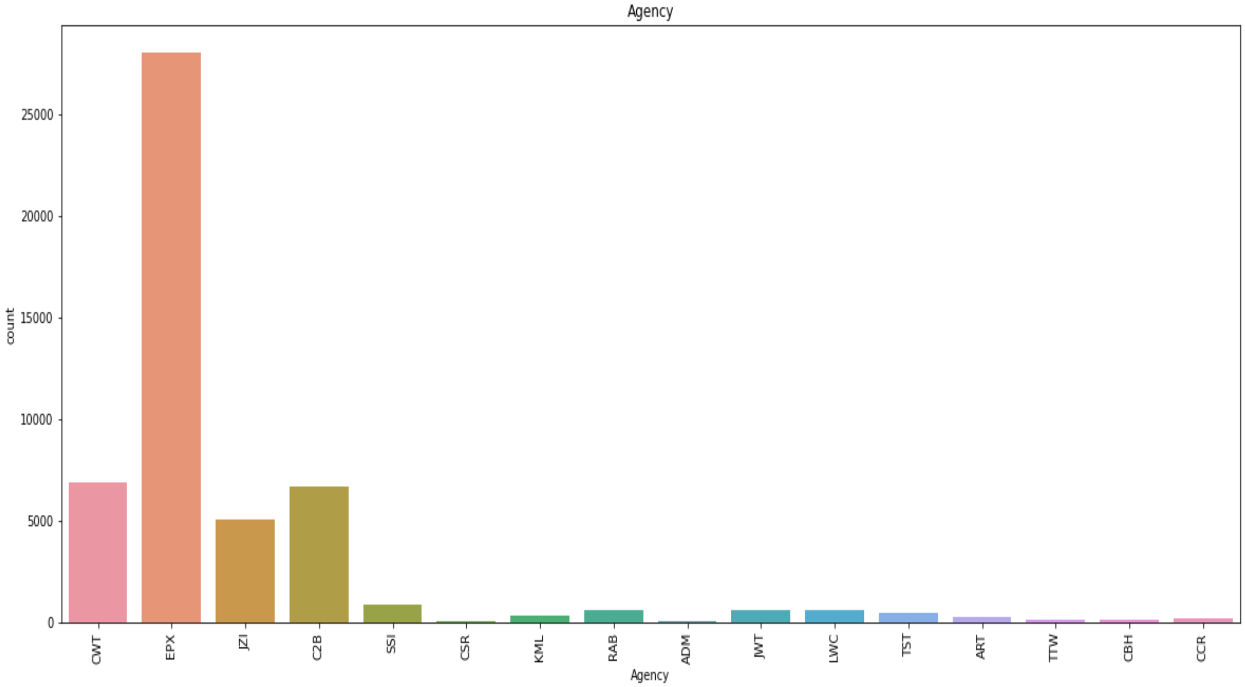
Using Seaborn.countplot() Basically it counts number of entries present



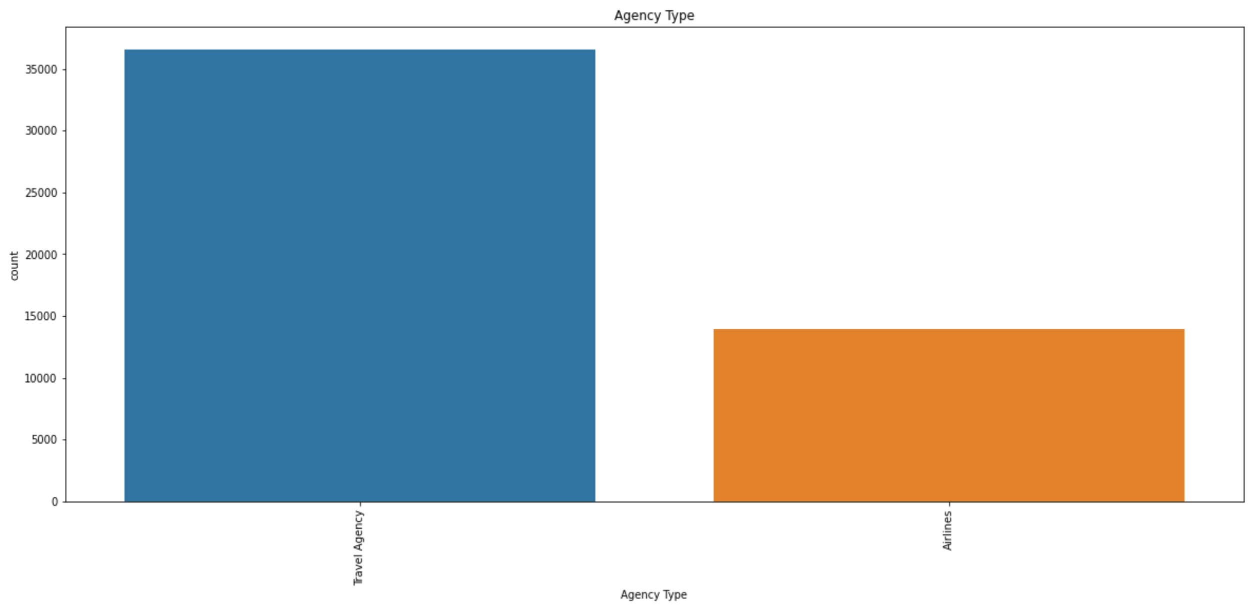
```
for i in df_cat:
  plt.figure(figsize=(20,8))
  sns.countplot(data=df_cat, x=i)
  plt.title(i)
  plt.xticks(rotation = 90)
  plt.show()
```

And Here are result,

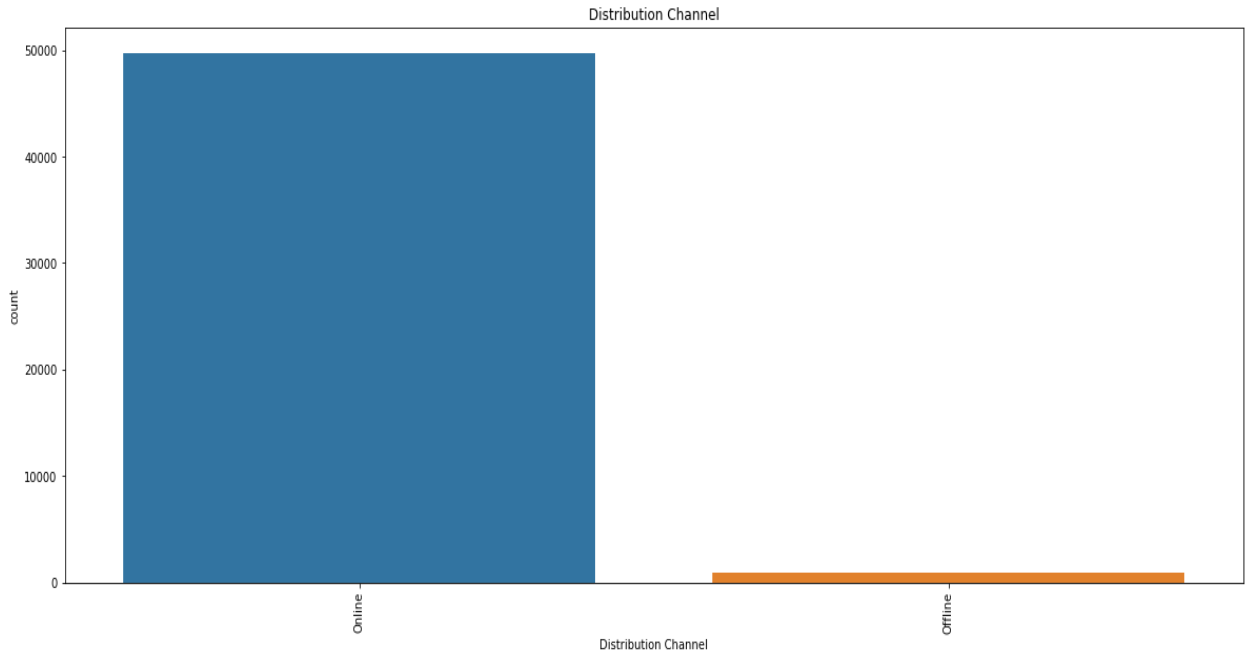
a. Agency



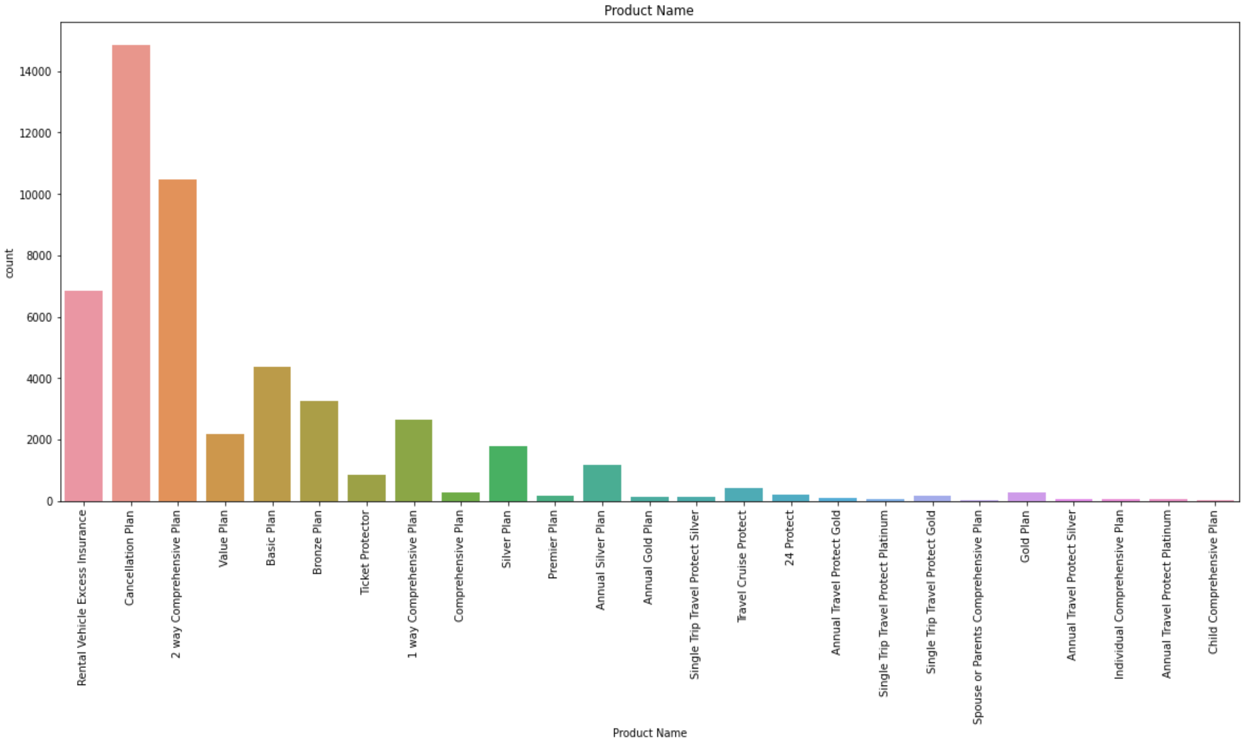
b. Agency Type



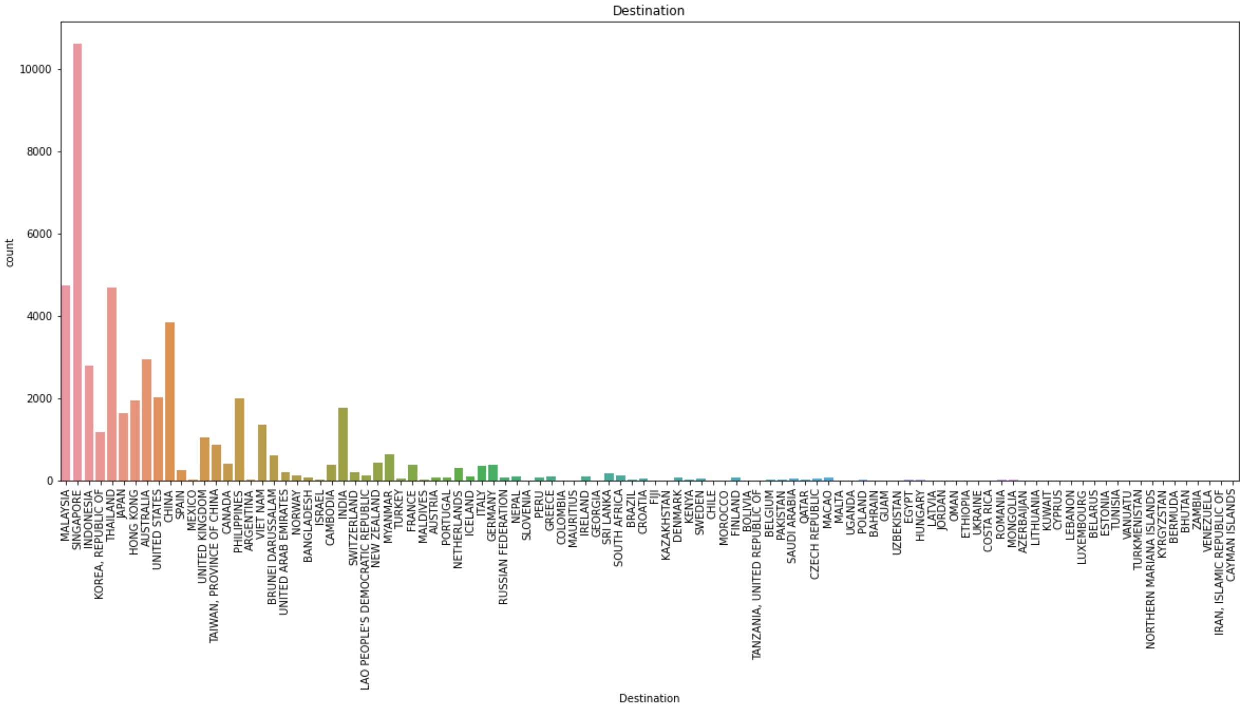
c. Distribution Channel



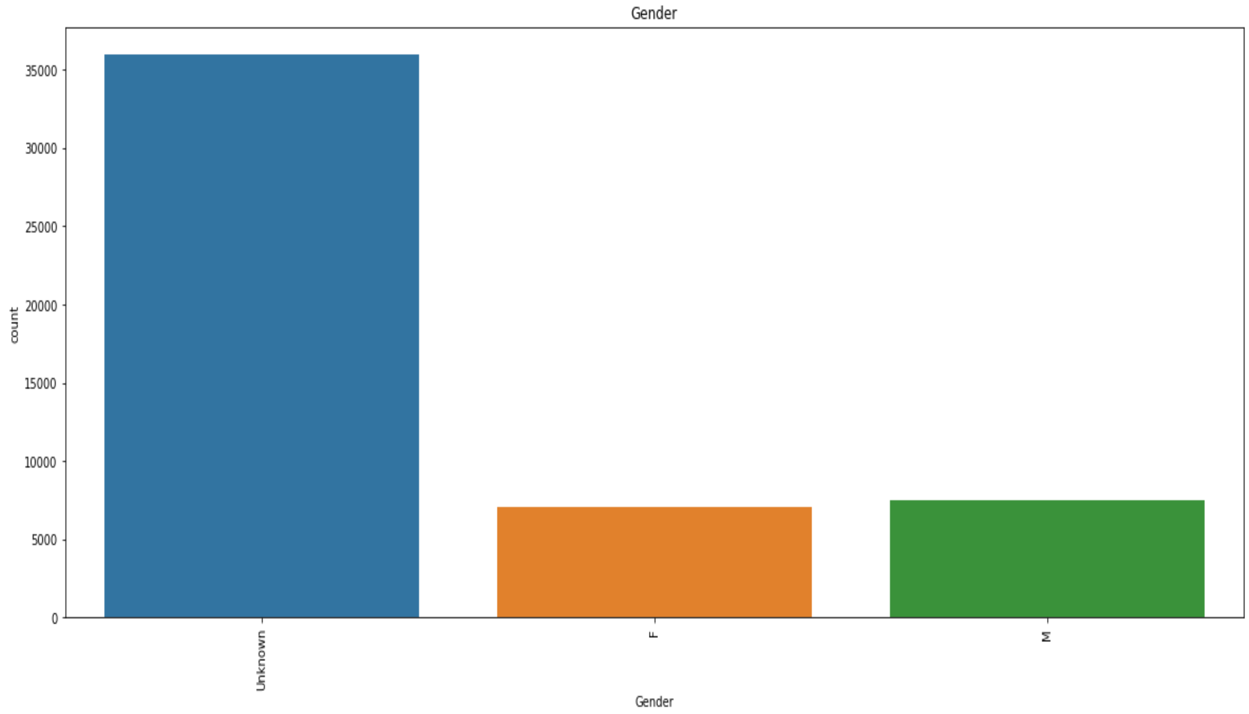
d. Product Name



e. Destination

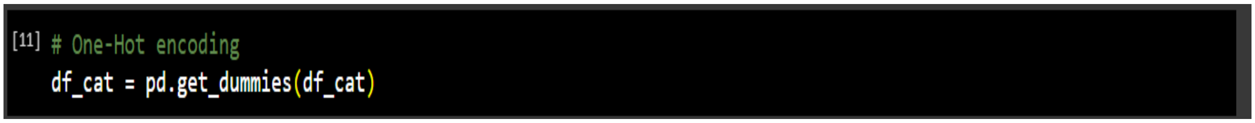


f. Gender



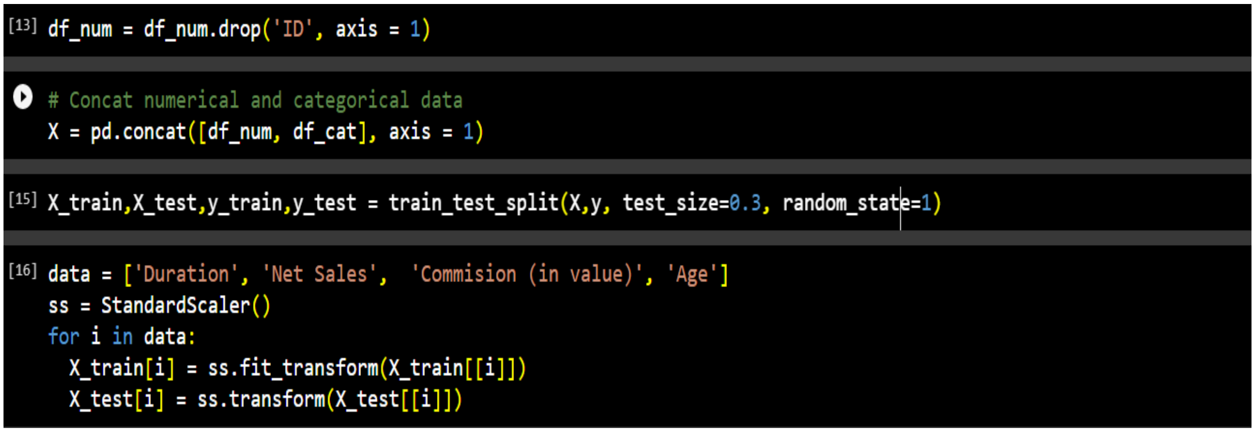
7. One-Hot Encoding

Here we use One-Hot Encoding not Label Ending because it ranges out from scale like take here example of Gender we have 3 types M, F and Unknown. So if we apply Label Encoding then it assign for example For M - 1, F - 2 and for Unknown - 3 which may be problematic when we apply activation Function. Like we use the tanh activation function in the program it has a range of -1 to 1.



```
[11] # One-Hot encoding
     df_cat = pd.get_dummies(df_cat)
```

8. Drop, Concat and Scaling



```
[13] df_num = df_num.drop('ID', axis = 1)
```

```
# Concat numerical and categorical data
X = pd.concat([df_num, df_cat], axis = 1)
```

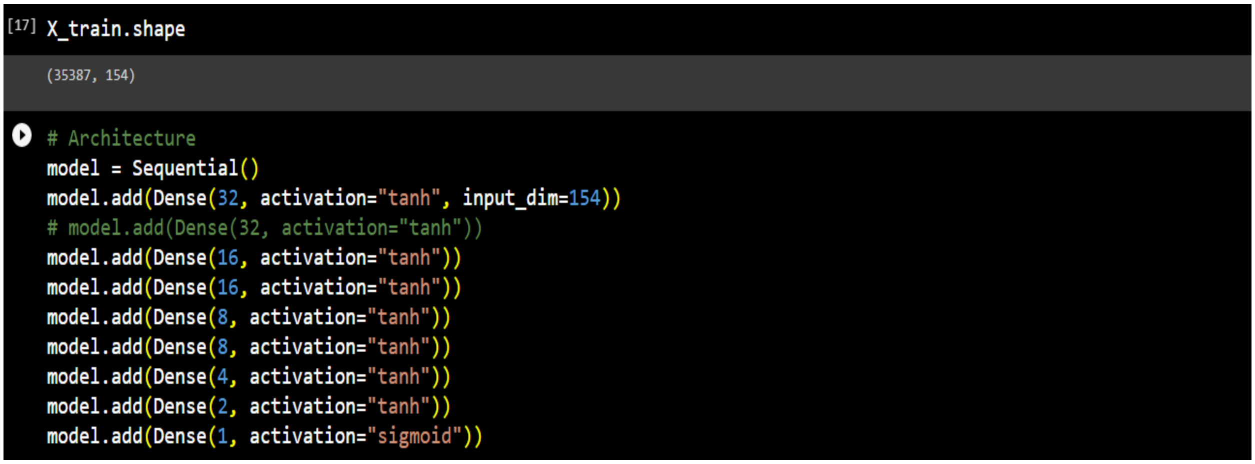
```
[15] X_train,X_test,y_train,y_test = train_test_split(X,y, test_size=0.3, random_state=1)
```

```
[16] data = ['Duration', 'Net Sales',  'Commision (in value)', 'Age']
     ss = StandardScaler()
     for i in data:
       X_train[i] = ss.fit_transform(X_train[[i]])
       X_test[i] = ss.transform(X_test[[i]])
```

Now here we drop out the ID column which is not necessary. Then merge numerical and categorical data then perform scaling to data like 'Duration', 'Net Sales', 'Commision (in value)', 'Age'.

Or before merging we can perform scaling to numerical data which is also the one way. But here we perform after merging data.

9. Architecture



```
[17] X_train.shape

     (35387, 154)
```

```
# Architecture
model = Sequential()
model.add(Dense(32, activation="tanh", input_dim=154))
# model.add(Dense(32, activation="tanh"))
model.add(Dense(16, activation="tanh"))
model.add(Dense(16, activation="tanh"))
model.add(Dense(8, activation="tanh"))
model.add(Dense(8, activation="tanh"))
model.add(Dense(4, activation="tanh"))
model.add(Dense(2, activation="tanh"))
model.add(Dense(1, activation="sigmoid"))
```

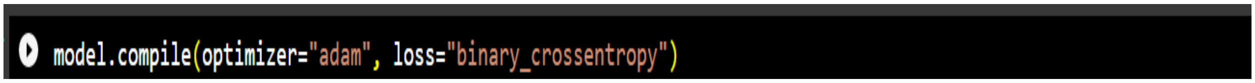
After Labeling and scaling stuff we get shape of `(35387, 154)` so we use 154 as a input dimension in the first layer of architecture.

Now here we use 8 layer including output layer

From which 1st layer contain of 32 neurons and activation function used tanh (Which is why we use One-Hot encoding)

2nd layer 16 neurons, 3rd layer 16 neurons, 4th layer 8 neuron, 5th layer 8 neuron, 6th layer 4 neuron, 7th layer 2 neuron

And the 8th layer which is the output layer, using one neuron with activation function of Sigmoid. Because here we have binary classification.



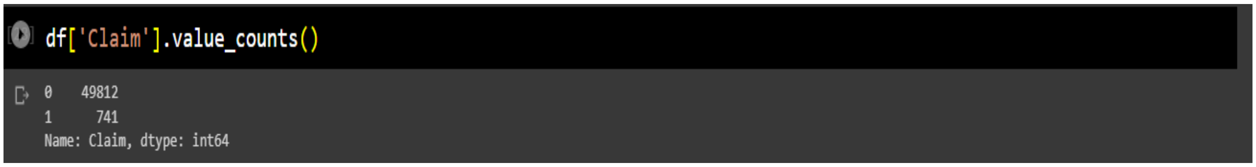
```
model.compile(optimizer="adam", loss="binary_crossentropy")
```

After that using Adam Optimizer and loss function for classification is binary cross entropy

10. Sampling

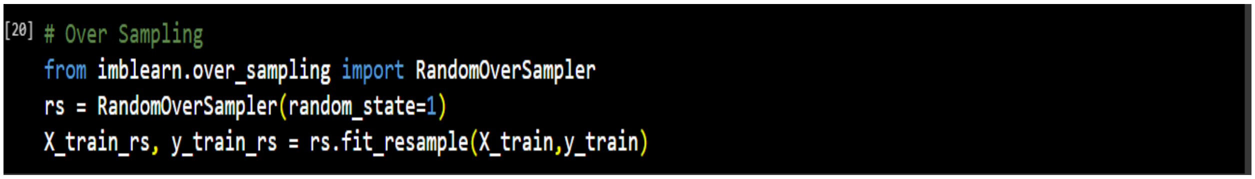
Now here target is highly unstable so here is data before sampling for 0 count is 49812

And for 1 count is 741



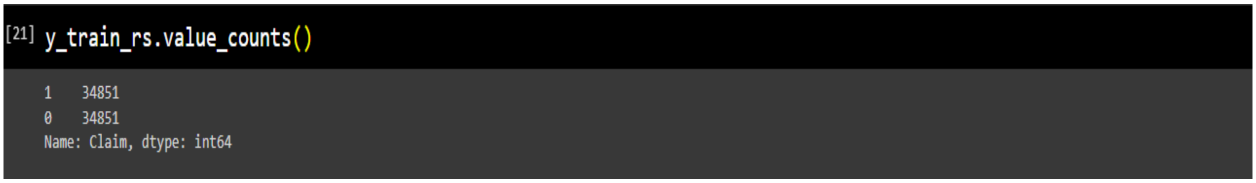
```
df['Claim'].value_counts()
```

```
0    49812
1      741
Name: Claim, dtype: int64
```



```
# Over Sampling
from imblearn.over_sampling import RandomOverSampler
rs = RandomOverSampler(random_state=1)
X_train_rs, y_train_rs = rs.fit_resample(X_train,y_train)
```

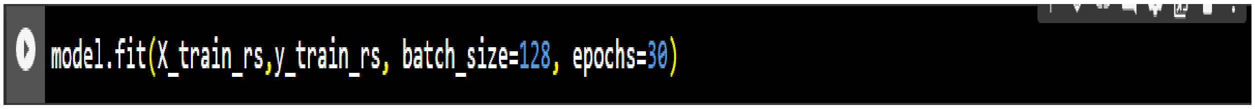
After Performing Over Sampling



```
y_train_rs.value_counts()
```

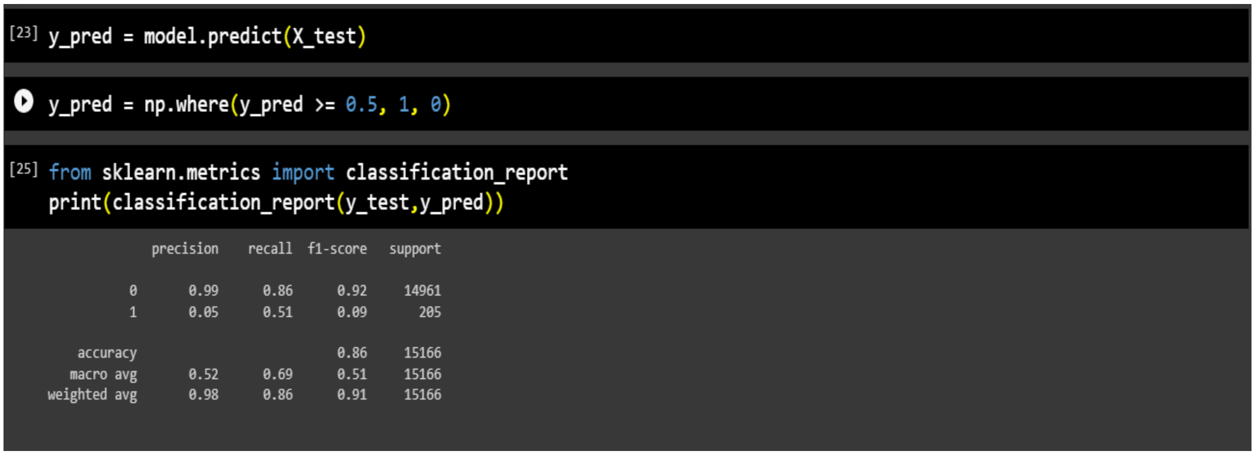
```
1    34851
0    34851
Name: Claim, dtype: int64
```

11. Finalizing



```
model.fit(X_train_rs,y_train_rs, batch_size=128, epochs=30)
```

Using batch size of 128 and epoch 30



```
[23] y_pred = model.predict(X_test)
```

```
y_pred = np.where(y_pred >= 0.5, 1, 0)
```

```
[25] from sklearn.metrics import classification_report
     print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.99      0.86      0.92     14961
           1       0.05      0.51      0.09       205

    accuracy                           0.86     15166
   macro avg       0.52      0.69      0.51     15166
weighted avg       0.98      0.86      0.91     15166
```