

```

import pandas as pd
import numpy as np
from imblearn.over_sampling import SMOTE

import seaborn as sns
import matplotlib.pyplot as plt
sns.set()

from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, classifi
from sklearn.metrics import roc_curve, auc

import xgboost as xgb
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import VotingClassifier

from google.colab import drive
drive.mount("/content/drive")

data = pd.read_csv("bs140513_032310.csv")
data.head(5)

```



Mounted at /content/drive



	step	customer	age	gender	zipcodeOri	
0	0	'C1093826151'	'4'	'M'	'28007'	'
1	0	'C352968107'	'2'	'M'	'28007'	'
2	0	'C2054744914'	'4'	'F'	'28007'	'N
3	0	'C1760612790'	'3'	'M'	'28007'	'
4	0	'C757503768'	'5'	'M'	'28007'	'

```
data.info()
```



```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 594643 entries, 0 to 594642
Data columns (total 10 columns):
#   Column                Non-Null Count  Dtype
---  -
0   step                  594643 non-null  int64
1   customer              594643 non-null  object
2   age                   594643 non-null  object
3   gender                594643 non-null  object
4   zipcodeOri            594643 non-null  object
5   merchant              594643 non-null  object
6   zipMerchant           594643 non-null  object
7   category              594643 non-null  object
8   amount                594643 non-null  float64

```

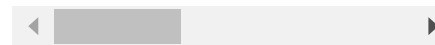
AssertionError X



You

Please explain the error:

AssertionError: View function



Gemini

Hypothesis

It looks like you might have defined the `home()` function twice, leading to Flask's complaint about overwriting an existing endpoint. The traceback indicates an `AssertionError` because the Flask app is attempting to map a view function to a route that already has a function assigned.

Suggested Changes



```

from flask_ngrok import run_w
from flask import Flask, requ
import pickle
import numpy as np

```

```

#Creating flask app
app = Flask(__name__)
run_with_ngrok(app)

```

```

# Load the model - This shoul
model = pickle.load(open('mod

```

```

@app.route('/') # Define the
def home():
    return render_template('i

```

```

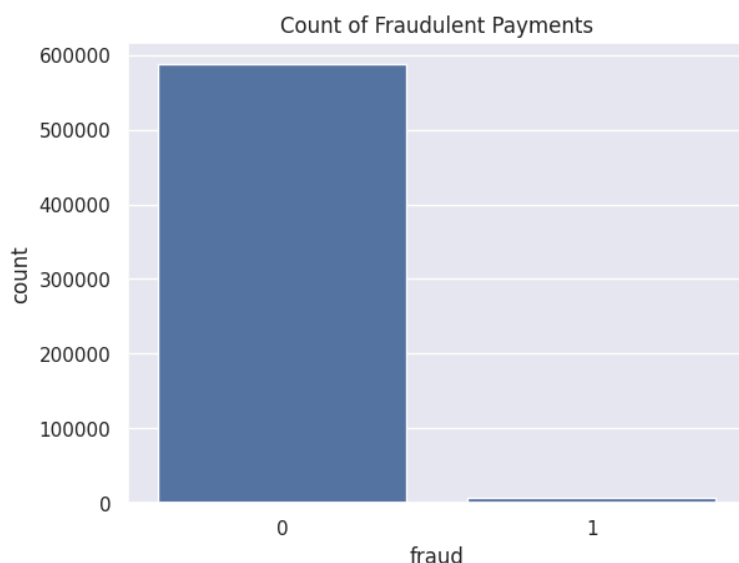
@app.route('/getprediction',m
def getprediction():

```

```
9    fraud      594643 non-null  int64
dtypes: float64(1), int64(2), object(7)
memory usage: 45.4+ MB
```

```
# Create two dataframes with fraud and non-fraud data
df_fraud = data.loc[data.fraud == 1]
df_non_fraud = data.loc[data.fraud == 0]
```

```
sns.countplot(x="fraud",data=data)
plt.title("Count of Fraudulent Payments")
plt.show()
print("Number of normal examples: ",df_non_fraud.fraud.
print("Number of fraudulent examples: ",df_fraud.fraud.
```



```
Number of normal examples: 587443
Number of fraudulent examples: 7200
```

```
pd.concat([df_fraud.groupby('category')['amount'].mean(),
data.groupby('category')['fraud'].mean()*100],
sort=False).sort_values(by=['Non-Fraudulent'])
```

```
input = [float(x) for x in input_data]
final_input = [np.array(input_data)]
prediction = model.predict(final_input)
```

```
return render_template('index.html', prediction=prediction)
```

```
if __name__ == '__main__':
    app.run()
```

[Use code with caution](#)

Explanation:

- Model Loading:** I've moved the model loading outside of any routes. This way, the model is loaded only once when the application starts, improving efficiency.
- Prediction Extraction:** I adjusted how you extract the prediction from `predict_proba` to directly get the probability of the positive class (fraudulent transaction).

class (fraudulent transaction).

Make sure you don't have another `home()` function defined elsewhere in your code. If you still encounter issues, double-check for any typos or accidental redefinitions.

Sources

- [mupadhya-ta/WebApp](#)
- [Abbey225/Loan-Default](#)

Rate this answer



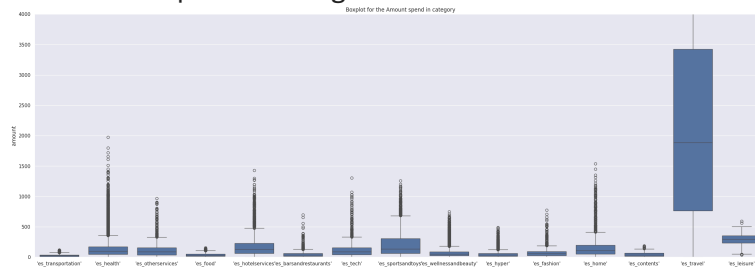


category	Fraudulent	Non-Fraudulent	Percentage
'es_transportation'	NaN	26.958187	
'es_food'	NaN	37.070405	
'es_hyper'	169.255429	40.037145	
'es_barsandrestaurants'	164.092667	41.145997	
'es_contents'	NaN	44.547571	
'es_wellnessandbeauty'	229.422535	57.320219	
'es_fashion'	247.008190	62.347674	
'es_leisure'	300.286878	73.230400	
'es_otherservices'	316.469605	75.685497	
'es_sportsandtoys'	345.366811	88.502738	
'es_tech'	415.274114	99.924638	
'es_health'	407.031338	103.737228	
'es_hotelservices'	421.823339	106.548545	
'es_home'	457.484834	113.338409	

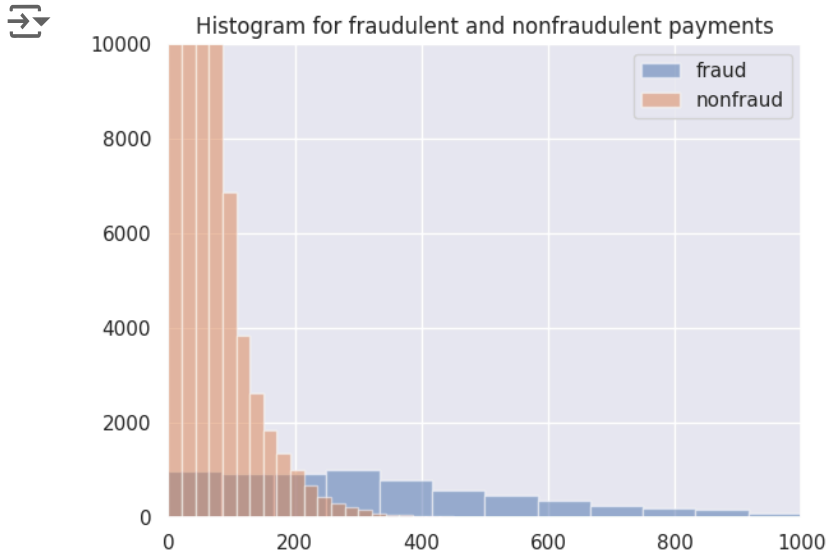
```
# Plot histograms of the amounts in fraud and non-fraud
plt.figure(figsize=(30,10))
sns.boxplot(x=data.category,y=data.amount)
plt.title("Boxplot for the Amount spend in category")
plt.ylim(0,4000)
plt.legend()
plt.show()
```



WARNING:matplotlib.legend:No artists with labels found in the legend. Legend has 1 entries.



```
# Plot histograms of the amounts in fraud and non-fraud
plt.hist(df_fraud.amount, alpha=0.5, label='fraud', bins=100)
plt.hist(df_non_fraud.amount, alpha=0.5, label='nonfraud', bins=100)
plt.title("Histogram for fraudulent and nonfraudulent payments")
plt.ylabel("Frequency")
plt.xlabel("Amount")
plt.legend()
plt.show()
```



```
print((data.groupby('age')['fraud'].mean()*100).reset_index())
```

	Age	Fraud Percent
7	'U'	0.594228
6	'6'	0.974826
5	'5'	1.095112
1	'1'	1.185254
3	'3'	1.192815
2	'2'	1.251401
4	'4'	1.293281
0	'0'	1.957586

✓ Data Preprocessing

```
print("Unique zipCodeOri values: ",data.zipcodeOri.nunique())
print("Unique zipMerchant values: ",data.zipMerchant.nunique())
```

```
Unique zipCodeOri values: 1
Unique zipMerchant values: 1
```

There are only one unique zipCode values so we will drop them.

```
# dropping zipcodeori and zipMerchant since they have
data_reduced = data.drop(['zipcodeOri','zipMerchant'],
```

```
data_reduced.columns
```

```
Index(['step', 'customer', 'age', 'gender',
       'merchant', 'category', 'amount',
       'fraud'],
      dtype='object')
```

```
# turning object columns type to categorical for easir
col_categorical = data_reduced.select_dtypes(include=
for col in col_categorical:
    data_reduced[col] = data_reduced[col].astype('cate
# categorical values ==> numeric values
data_reduced[col_categorical] = data_reduced[col_cate
data_reduced.head(5)
```

```

step  customer  age  gender  merchant  category
0      0      210   4      2       30       12
1      0     2753   2      2       30       12
2      0     2285   4      1       18       12
3      0     1650   3      2       30       12
4      0     3585   5      2       30       12
```

Let's define our independent variable (X) and dependant/target variable y

```
X = data_reduced.drop(['fraud'],axis=1)
y = data['fraud']
print(X.head(),"\n")
print(y.head())
```

```

step  customer  age  gender  merchant  category
0      0      210   4      2       30       12
1      0     2753   2      2       30       12
2      0     2285   4      1       18       12
3      0     1650   3      2       30       12
4      0     3585   5      2       30       12

0      0
1      0
2      0
```

```
3    0
4    0
Name: fraud, dtype: int64
```

```
y[y==1].count()
```

```
7200
```

Over Sampling using SMOTE

```
sm = SMOTE(random_state=42)
X_res, y_res = sm.fit_resample(X, y)
y_res = pd.DataFrame(y_res, columns=['fraud'])
print(y_res['fraud'].value_counts())
```

```
fraud
0    587443
1    587443
Name: count, dtype: int64
```

I will do a train test split for measuring the performance.

```
X_train, X_test, y_train, y_test = train_test_split(X,
```

```
# %% Function for plotting ROC_AUC curve
```

```
def plot_roc_auc(y_test, preds):
    ...
    Takes actual and predicted(probabilities) as input
    Operating Characteristic (ROC) curve
    ...
    fpr, tpr, threshold = roc_curve(y_test, preds)
    roc_auc = auc(fpr, tpr)
    plt.title('Receiver Operating Characteristic')
    plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc_auc)
    plt.legend(loc = 'lower right')
    plt.plot([0, 1], [0, 1], 'r--')
    plt.xlim([0, 1])
    plt.ylim([0, 1])
    plt.ylabel('True Positive Rate')
    plt.xlabel('False Positive Rate')
    plt.show()
```

```
# The base score should be better than predicting always
print("Base accuracy score we must beat is: ",
      df_non_fraud.fraud.count() / np.add(df_non_fraud.fraud,
```

```
Base accuracy score we must beat is: 98.789189486
```

KNN Clasification

```
# %% K-ello Neighbors

knn = KNeighborsClassifier(n_neighbors=5,p=1)

knn.fit(X_train,y_train)
y_pred = knn.predict(X_test)

print("Classification Report for K-Nearest Neighbours:
print("Confusion Matrix of K-Nearest Neighbours: \n", c
plot_roc_auc(y_test, knn.predict_proba(X_test)[: ,1])
```

```
→ /usr/local/lib/python3.10/dist-packages/sklearn/ne
```

```
    return self._fit(X, y)
```

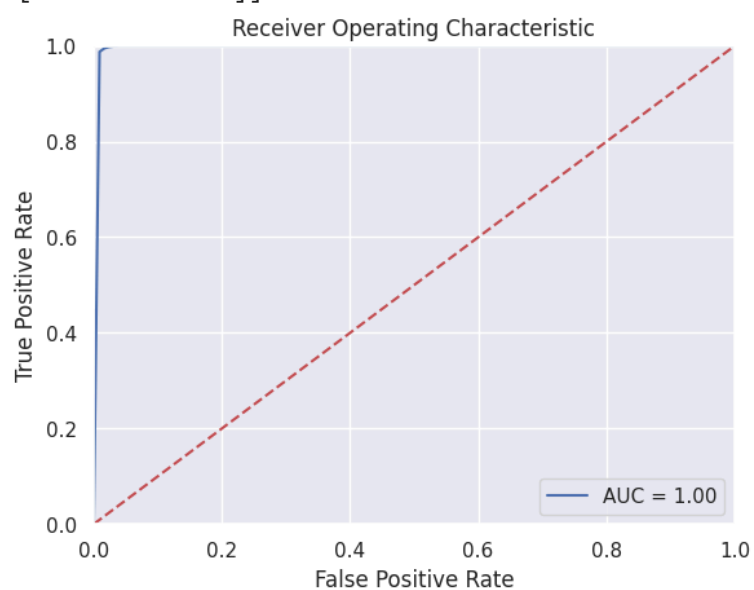
Classification Report for K-Nearest Neighbours:

	precision	recall	f1-score	sup
0	1.00	0.98	0.99	176
1	0.98	1.00	0.99	176
accuracy			0.99	352
macro avg	0.99	0.99	0.99	352
weighted avg	0.99	0.99	0.99	352

Confusion Matrix of K-Nearest Neighbours:

```
[[171999  4234]
```

```
[   362 175871]]
```



Random Forest Classifier


```
# %% Random Forest Classifier
```

```
rf_clf = RandomForestClassifier(n_estimators=100,max_c
                                verbose=1,class_weight
```

```
rf_clf.fit(X_train,y_train)
y_pred = rf_clf.predict(X_test)
```

```
print("Classification Report for Random Forest Classif
print("Confusion Matrix of Random Forest Classifier: \
plot_roc_auc(y_test, rf_clf.predict_proba(X_test)[: ,1]
```



```
<ipython-input-22-e57e8f380b47>:6: DataConversion\
rf_clf.fit(X_train,y_train)
```

```
[Parallel(n_jobs=1)]: Done 49 tasks      | elapsed
```

```
[Parallel(n_jobs=1)]: Done 49 tasks      | elapsed
```

```
Classification Report for Random Forest Classifier
```

```
precision    recall  f1-score   sup
```

```
0           0.99      0.97      0.98      176
```

```
1           0.97      0.99      0.98      176
```

```
accuracy                0.98      35%
```

```
macro avg              0.98      0.98      0.98      35%
```

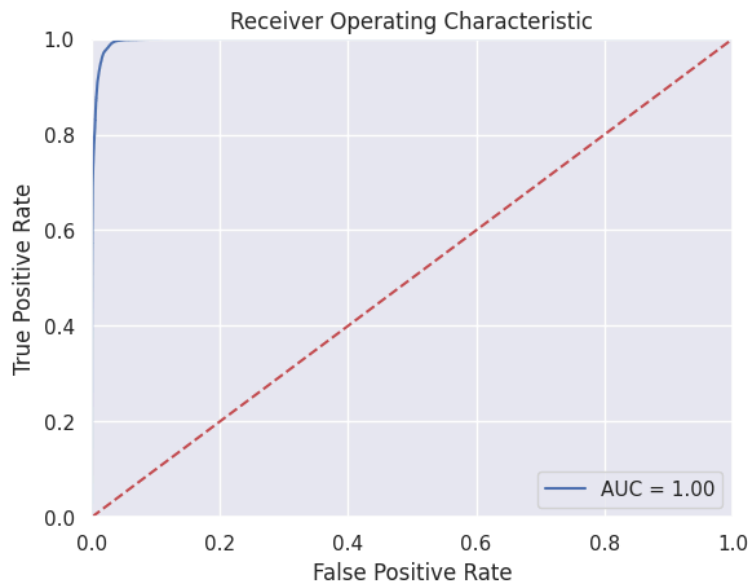
```
weighted avg          0.98      0.98      0.98      35%
```

```
Confusion Matrix of Random Forest Classifier:
```

```
[[170106  6127]
```

```
[ 1079 175154]]
```

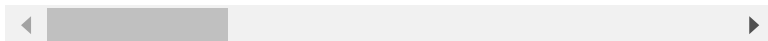
```
[Parallel(n_jobs=1)]: Done 49 tasks      | elapsed
```



✓ Deployment

```
model=rf_clf.fit(X_train,y_train)
```

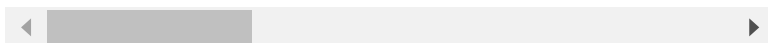
```
↔ <ipython-input-23-62ead54a5d96>:1: DataConversionWarning:
  model=rf_clf.fit(X_train,y_train)
[Parallel(n_jobs=1)]: Done 49 tasks | elapsed
```



```
new_data=np.array([[0,2500,6,1,18,12,55.55]]).reshape((1,7))
```

```
prediction = model.predict_proba(new_data)[:,-1][0]
print("\033[1m This new transaction has a {:.2%}".format(prediction))
```

```
↔ This new transaction has a 0.65% chance of being fraudulent.
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:464: UserWarning:
  warnings.warn('Model does not have a predict_proba attribute')
[Parallel(n_jobs=1)]: Done 49 tasks | elapsed
```



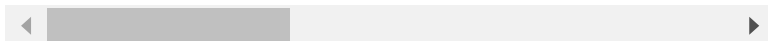
```
import pickle
```

Start coding or [generate](#) with AI.

Install ngrok

```
!pip install flask-ngrok
```

```
↔ Collecting flask-ngrok
  Downloading flask_ngrok-0.0.25-py3-none-any.whl (10.5 kB)
Requirement already satisfied: Flask>=0.8 in /usr/local/lib/python3.10/dist-packages (2.0.3)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (2.31.0)
Requirement already satisfied: Werkzeug>=2.2.2 in /usr/local/lib/python3.10/dist-packages (3.0.2)
Requirement already satisfied: Jinja2>=3.0 in /usr/local/lib/python3.10/dist-packages (3.1.2)
Requirement already satisfied: itsdangerous>=2.0 in /usr/local/lib/python3.10/dist-packages (2.1.2)
Requirement already satisfied: click>=8.0 in /usr/local/lib/python3.10/dist-packages (8.1.7)
Requirement already satisfied: charset-normalizer<4, >=2.0 in /usr/local/lib/python3.10/dist-packages (3.3.2)
Requirement already satisfied: idna<4, >=2.5 in /usr/local/lib/python3.10/dist-packages (3.6)
Requirement already satisfied: urllib3<3, >=1.21.1 in /usr/local/lib/python3.10/dist-packages (2.2.3)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (2024.7.4)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages (2.1.5)
Installing collected packages: flask-ngrok
Successfully installed flask-ngrok-0.0.25
```



```
from flask_ngrok import run_with_ngrok
from flask import Flask, request, render_template
```

✓ Create Flask Application

```
#Creating flask app
app = Flask(__name__)
run_with_ngrok(app)
f1=open('model.pkl','wb')
pickle.dump(model,f1)
f1.close()

model=pickle.load(open('model.pkl','rb'))
```

Gemini access in Colab is available for a limited time for unsubscribed users.

[Upgrade to Colab Pro](#)

Enter a prompt here



0 / 400

Responses may display inaccurate or offensive information that doesn't represent Google's views. [Learn more](#)