1.) Import the Credit Card Fraud Data From CCLE¶

In [38]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
```

In [39]:
```
df = pd.read_csv("/Users/snehilshandilya/Desktop/fraudTest_2.csv")
```

In [40]:
```
df.head()
```

Out[40]:

	Unnamed: 0	trans_date_trans_time	cc_num	merchant	category	amt	first	last	gender	street	...	la
0	0	2020-06-21 12:14:25	2291163933867244	fraud_Kirlin and Sons	personal_care	2.86	Jeff	Elliott	M	351 Darlene Green	...	33.965
1	1	2020-06-21 12:14:33	3573030041201292	fraud_Sporer-Keebler	personal_care	29.84	Joanne	Williams	F	3638 Marsh Union	...	40.320
2	2	2020-06-21 12:14:53	3598215285024754	fraud_Swaniawski, Nitzsche and Welch	health_fitness	41.28	Ashley	Lopez	F	9333 Valentine Point	...	40.672
3	3	2020-06-21 12:15:15	3591919803438423	fraud_Haley Group	misc_pos	60.05	Brian	Williams	M	32941 Krystal Mill Apt. 552	...	28.569
4	4	2020-06-21 12:15:17	3526826139003047	fraud_Johnston-Casper	travel	3.19	Nathan	Massey	M	5783 Evan Roads Apt. 465	...	44.252

5 rows × 23 columns

2.) Select four columns to use as features (one must be trans_date_trans)

In [41]:
```
df_select = df[["trans_date_trans_time", "category", "amt", "city_pop", "is_fraud"]]
df_select.head()
```

Out[41]:

	trans_date_trans_time	category	amt	city_pop	is_fraud
0	2020-06-21 12:14:25	personal_care	2.86	333497	0
1	2020-06-21 12:14:33	personal_care	29.84	302	0
2	2020-06-21 12:14:53	health_fitness	41.28	34496	0
3	2020-06-21 12:15:15	misc_pos	60.05	54767	0
4	2020-06-21 12:15:17	travel	3.19	1126	0

3.) Create a unique variable out of trans_date.

In [42]:
```
df["trans_date_trans_time"] = pd.to_datetime(df["trans_date_trans_time"])
```

In [43]:
```
df_select["time_var"] = [i.hour for i in df["trans_date_trans_time"]]
```

```
/var/folders/sv/s309_3dd79s_59j12prhgcd00000gn/T/ipykernel_22576/1545706583.py:1: SettingWithCopyWarnin
g:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.
html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.h
tml#returning-a-view-versus-a-copy)
  df_select["time_var"] = [i.hour for i in df["trans_date_trans_time"]]
```

In [44]:
```
df_select["time_var"]
```

Out[44]:
```
0         12
1         12
2         12
3         12
4         12
          ..
555714    23
555715    23
555716    23
555717    23
555718    23
Name: time_var, Length: 555719, dtype: int64
```

In [45]:
```
dummies = pd.get_dummies(df_select["category"])
X = pd.concat([dummies, df_select[["amt", "city_pop", "time_var"]]], axis = 1)
Y = df_select["is_fraud"]
```

In [46]:
```
X.head()
```

Out[46]:

grocery_net	grocery_pos	health_fitness	home	kids_pets	misc_net	misc_pos	personal_care	shopping_net	shopping_pos	travel	amt	cit
0	0	0	0	0	0	0	1	0	0	0	2.86	3
0	0	0	0	0	0	0	1	0	0	0	29.84	
0	0	1	0	0	0	0	0	0	0	0	41.28	
0	0	0	0	0	0	1	0	0	0	0	60.05	
0	0	0	0	0	0	0	0	0	0	1	3.19	

In [47]:
```
from sklearn import preprocessing
min_max_scaler = preprocessing.MinMaxScaler(feature_range = (0,1))
X = min_max_scaler.fit_transform(X)
```

4.) Splitting the data into training and testing set

In [48]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X,Y, test_size = 0.3)
```

5.) Making three sets of training data

a) Oversample

In [49]:
```
pip install imblearn
```

```
Requirement already satisfied: imblearn in ./opt/anaconda3/lib/python3.9/site-packages (0.0)
Requirement already satisfied: imbalanced-learn in ./opt/anaconda3/lib/python3.9/site-packages (from im
blearn) (0.10.1)
Requirement already satisfied: threadpoolctl>=2.0.0 in ./opt/anaconda3/lib/python3.9/site-packages (fro
m imbalanced-learn->imblearn) (2.2.0)
Requirement already satisfied: scipy>=1.3.2 in ./opt/anaconda3/lib/python3.9/site-packages (from imbala
nced-learn->imblearn) (1.7.3)
Requirement already satisfied: joblib>=1.1.1 in ./opt/anaconda3/lib/python3.9/site-packages (from imbal
anced-learn->imblearn) (1.2.0)
Requirement already satisfied: scikit-learn>=1.0.2 in ./opt/anaconda3/lib/python3.9/site-packages (from
imbalanced-learn->imblearn) (1.0.2)
Requirement already satisfied: numpy>=1.17.3 in ./opt/anaconda3/lib/python3.9/site-packages (from imbal
anced-learn->imblearn) (1.21.5)
Note: you may need to restart the kernel to use updated packages.
```

In [50]:
```
from imblearn.over_sampling import RandomOverSampler
```

In [51]:
```
ros = RandomOverSampler(random_state = 0)
ros.fit(X_train,Y_train)
X_resampled_over, Y_resampled_over = ros.fit_resample(X_train,Y_train)
```

b) Undersampling

In [52]:
```
from imblearn.under_sampling import RandomUnderSampler
```

In [53]:
```
rus = RandomUnderSampler(random_state = 0)
rus.fit(X_train,Y_train)
X_resampled_under, Y_resampled_under = rus.fit_resample(X_train,Y_train)
```

c) SMOTE

In [54]:
```
from imblearn.over_sampling import SMOTE
```

In [55]:
```
oversample = SMOTE()
X_smote, Y_smote = oversample.fit_resample(X_train,Y_train)
```

Fitting regression models

In [56]:
```
from sklearn.linear_model import LogisticRegression
```

In [57]:
```
#Fitting for oversampled data

log_reg_over = LogisticRegression().fit(X_resampled_over,Y_resampled_over)
```

In [58]:
```
#Fitting for undersampled data

log_reg_under = LogisticRegression().fit(X_resampled_under,Y_resampled_under)
```

In [59]:
```
#Fitting for SMOTE

log_reg_smote =  LogisticRegression().fit(X_smote,Y_smote)
```

Testing the models

a) Oversampled Data

In [63]:
```
# Get predictions
Y_pred = log_reg_over.predict(X_test)

# Evaluate the model performance using accuracy score
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(Y_test, Y_pred)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.9078372801650711
```

b) Undersampled Data

In [64]:
```
Y_pred1 = log_reg_under.predict(X_test)

# Evaluate the model performance using accuracy score
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(Y_test, Y_pred1)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.7463830706110991
```

c) SMOTE

In [65]:
```
Y_pred2 = log_reg_smote.predict(X_test)

# Evaluate the model performance using accuracy score
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(Y_test, Y_pred2)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.9063257275846349
```

On the basis of the accuracy of the three models, we see that SMOTE and Oversampled data performs the best in Out of Sample Metrics with an accuracy score of approximately 90%. On the other hand, the undersampled data is the least accurate with an accuracy score of 74%

Selecting two features

In [66]:
```
df_select[df_select["is_fraud"] ==1][["amt", "city_pop"]]
```

Out[66]:

	amt	city_pop
1685	24.84	23
1767	780.52	1306
1781	620.33	1306
1784	1077.69	71335
1857	842.65	23
...	...	...
517197	1041.51	14462
517274	868.09	14462
517341	1039.42	14462
517529	289.27	14462
517571	766.38	14462

2145 rows × 2 columns

In [68]:
```
import matplotlib.pyplot as plt
```

Plotting the two classes before SMOTE

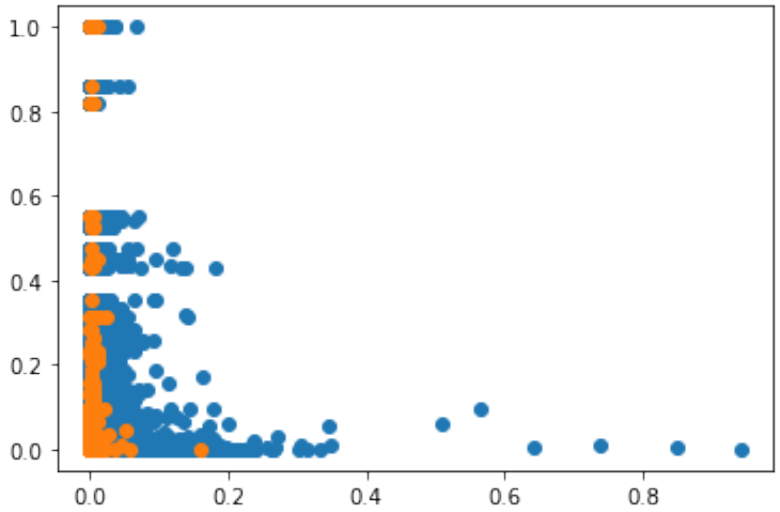
In [113]:
```
data_temp = pd.concat([X_train, Y_train], axis=1)

print(data_temp.columns)
```

```
Index([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 'is_fraud'], dtype='object')
```

In [115]:
```
plt.scatter(data_temp[data_temp["is_fraud"] == 0][14], data_temp[data_temp["is_fraud"] == 0][15])
plt.scatter(data_temp[data_temp["is_fraud"] == 1][14], data_temp[data_temp["is_fraud"] == 1][15])

plt.show()
```



Plotting the two classes after SMOTE

In [117]:
```
X_smote_new = pd.DataFrame(X_smote)
Y_smote_new = pd.DataFrame(Y_smote)
```

In [121]:
```
data_temp_new = pd.concat([X_smote_new, Y_smote_new], axis = 1)
```

In [122]:
```
plt.scatter(data_temp_new[data_temp_new["is_fraud"] == 0][14], data_temp_new[data_temp_new["is_fraud"] =
plt.scatter(data_temp_new[data_temp_new["is_fraud"] == 1][14], data_temp_new[data_temp_new["is_fraud"] =

plt.show()
```