```
import pandas as pd
import numpy as np
from google.colab import files
uploaded = files.upload()
```

Choose Files train.csv

- **train.csv**(text/csv) - 61194 bytes, last modified: 10/23/2023 - 100% done

```
Saving train.csv to train (1).csv
```

```
df = pd.read_csv('train.csv')
df.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S

```
df.duplicated()
```

```
0      False
1      False
2      False
3      False
4      False
       ...
886    False
887    False
888    False
889    False
890    False
Length: 891, dtype: bool
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  891 non-null    int64
 1   Survived     891 non-null    int64
 2   Pclass       891 non-null    int64
 3   Name         891 non-null    object
 4   Sex          891 non-null    object
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64
 7   Parch        891 non-null    int64
 8   Ticket       891 non-null    object
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object
 11  Embarked     889 non-null    object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

```
cat_col = [col for col in df.columns if df[col].dtype == 'object']
print('Categorical columns :',cat_col)
num_col = [col for col in df.columns if df[col].dtype != 'object']
print('Numerical columns :',num_col)
```

```
Categorical columns : ['Name', 'Sex', 'Ticket', 'Cabin', 'Embarked']
Numerical columns : ['PassengerId', 'Survived', 'Pclass', 'Age', 'SibSp', 'Parch', 'Fare']
```

```
df[cat_col].nunique()
```

```
Name        891
Sex           2
Ticket      681
Cabin       147
Embarked      3
dtype: int64
```

```
df['Ticket'].unique()[:50]
```

```
array(['A/5 21171', 'PC 17599', 'STON/O2. 3101282', '113803', '373450',
       '330877', '17463', '349909', '347742', '237736', 'PP 9549',
       '113783', 'A/5. 2151', '347082', '350406', '248706', '382652',
       '244373', '345763', '2649', '239865', '248698', '330923', '113788',
       '347077', '2631', '19950', '330959', '349216', 'PC 17601',
       'PC 17569', '335677', 'C.A. 24579', 'PC 17604', '113789', '2677',
       'A./5. 2152', '345764', '2651', '7546', '11668', '349253',
       'SC/Paris 2123', '330958', 'S.C./A.4. 23567', '370371', '14311',
       '2662', '349237', '3101295'], dtype=object)
```

```
df1 = df.drop(columns=['Name','Ticket'])
df1.shape
```

```
(891, 10)
```

```
round((df1.isnull().sum()/df1.shape[0])*100,2)
```

```
PassengerId     0.00
Survived        0.00
Pclass          0.00
Sex             0.00
Age            19.87
SibSp           0.00
Parch           0.00
Fare            0.00
Cabin          77.10
Embarked        0.22
dtype: float64
```

```
df2 = df1.drop(columns='Cabin')
df2.dropna(subset=['Embarked'], axis=0, inplace=True)
df2.shape
```

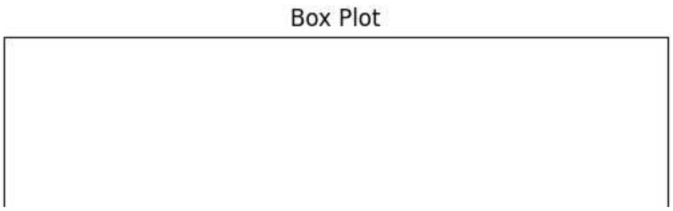
```
(889, 9)
```

```
df3 = df2.fillna(df2.Age.mean())
df3.isnull().sum()
```

```
PassengerId    0
Survived       0
Pclass         0
Sex            0
Age            0
SibSp          0
Parch          0
Fare           0
Embarked       0
dtype: int64
```

```
import matplotlib.pyplot as plt

plt.boxplot(df3['Age'], vert=False)
plt.ylabel('Variable')
plt.xlabel('Age')
plt.title('Box Plot')
plt.show()
```



```
mean = df3['Age'].mean()
std = df3['Age'].std()

lower_bound = mean - std*2
upper_bound = mean + std*2

print('Lower Bound :',lower_bound)
print('Upper Bound:',upper_bound)

df4 = df3[(df3['Age'] >= lower_bound)
& (df3['Age'] <= upper_bound)]
```

```
Lower Bound : 3.7054001079256587
Upper Bound: 55.57878528533277
```

```
X = df3[['Pclass','Sex','Age','SibSp','Parch','Fare','Embarked']]
Y = df3['Survived']
```

```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler(feature_range=(0,1))

num_col_ = [col for col in X.columns if X [col].dtype != 'object']
x1 = X

x1[num_col_] = scaler.fit_transform(x1[num_col_])
x1.head()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-15-209de419448c> in <cell line: 5>()
      3 scaler = MinMaxScaler(feature_range=(0,1))
      4 
----> 5 num_col_ = [col for col in X.columns if X [col].dtype != 'object']
      6 x1 = X
      7 

NameError: name 'X' is not defined
```

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import confusion_matrix, classification_report

from google.colab import files
uploaded= files.upload()
```

Choose Files creditcard.csv

- **creditcard.csv**(text/csv) - 532469 bytes, last modified: 10/23/2023 - 100% done

```
Saving creditcard.csv to creditcard (2).csv
```

```
data = pd.read_csv('creditcard.csv')
print(data.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 999 entries, 0 to 998
Data columns (total 1 columns):
 #   Column
---  ------
 0   Time,"V1","V2","V3","V4","V5","V6","V7","V8","V9","V10","V11","V12","V13","V14","V15","V16","V17","V18","V19","V20","V21","V22","V
dtypes: object(1)
```

```
memory usage: 7.9+ KB
None
```

```
print(data.columns)
```

```
Index(['Time,"V1","V2","V3","V4","V5","V6","V7","V8","V9","V10","V11","V12","V13","V14","V15","V16","V17","V18","V19","V20","V21","V22"
```

```
data ['normAmount'] = StandardScaler().fit_transform(np.array(data['Amount']).reshape(-1, 1))
data = data.drop(['Time', 'Amount'], axis = 1)
data['Class'].value_counts()
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
/usr/local/lib/python3.10/dist-packages/pandas/core/indexes/base.py in get_loc(self, key, method, tolerance)
   3801             try:
-> 3802                 return self._engine.get_loc(casted_key)
   3803             except KeyError as err:

                                  4 frames
pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_item()

pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_item()

KeyError: 'Amount'

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call last)
/usr/local/lib/python3.10/dist-packages/pandas/core/indexes/base.py in get_loc(self, key, method, tolerance)
   3802                 return self._engine.get_loc(casted_key)
   3803             except KeyError as err:
-> 3804                 raise KeyError(key) from err
   3805             except TypeError:
   3806                 # If we have a listlike key,  check indexing error will raise
```

```
from sklearn.model_selection import train_test_split

X = data.drop('Class', axis=1)
y = data['Class']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)

print("Number of transactions in X_train dataset: ", X_train.shape[0])
print("Number of transactions in y_train dataset: ", y_train.shape[0])
print("Number of transactions in X_test dataset: ", X_test.shape[0])
print("Number of transactions in y_test dataset: ", y_test.shape[0])
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-23-7de74c1a2160> in <cell line: 3>()
      1 from sklearn.model_selection import train_test_split
      2
----> 3 X = data.drop('Class', axis=1)
      4 y = data['Class']
      5

                                  5 frames
/usr/local/lib/python3.10/dist-packages/pandas/core/indexes/base.py in drop(self, labels, errors)
   6932         if mask.any():
   6933             if errors != "ignore":
-> 6934                 raise KeyError(f"{list(labels[mask])} not found in axis")
   6935             indexer = indexer[~mask]
   6936         return self.delete(indexer)
```

```
lr = LogisticRegression()

lr.fit(X_train, y_train.ravel())
lr.fit(X_train, y_train.ravel())

prediction = lr.predict(X_test)

print(classification_report(y_test, prediction))
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-24-5f9627479a59> in <cell line: 3>()
      1 lr = LogisticRegression()
      2 
----> 3 lr.fit(X_train, y_train.ravel())
      4 
      5 prediction = lr.predict(X_test)

NameError: name 'X_train' is not defined
```

```
print("Before OverSampling, counts of label '1' : {}".format(sum(y_train == 1)))
print("Before OverSampling, counts of label '0' : {} \n".format(sum(y_train == 0)))

from imblearn.over_sampling import SMOTE
sm = SMOTE(random_state = 2)
X_train_res, y_train_res = (X_train, y_train.ravel())


print('After OverSampling, the shape of train_X: {}'. format(X_train_res.shape))
print('After OverSampling, the shape of train_y: {} \n'. format(y_train_res.shape))

print("After OverSampling, counts of label '1': {}".format(sum(y_train_res == 1)))
print("After OverSampling, counts of label '0': {}".format(sum(y_train_res == 0)))
```

```
lr  = LogisticRegression()
lr.fit(X_train, y_train.ravel())

predictions = lr.predict(X_test)

print(classification_report(y_test,predictions))
```

```
print("Before OverSampling, counts of label '1': {}".format(sum(y_train == 1)))
print("Before OverSampling, counts of label '0': {} \n".format(sum(y_train == 0)))

from imblearn.over_sampling import SMOTE
sm = SMOTE(random_state = 2)
X_train_res, y_train_res = (X_train, y_train.ravel())

print('After OverSampling, the shape of train_X: {}'.format(X_train_res.shape))
print('After OverSampling, the shape of train_y: {}\n'.format(X_train_res.shape))

print("After OverSampling, counts of label '1': {}". format(sum(y_train_res == 1)))
print("After OverSampling, counts of label '0': {}". format(sum(y_train_res == 0)))
```

```
lr1 = LogisticRegression()
lr1.fit(X_train_res, y_train_res.ravel())
prediction = lr1.predict(X_test)

print(classification_report(y_test, prediction))
```

```
print("Before Undersampling, counts of label '1': {}".format(sum(y_train == 1 )))
print("Before Undersampling, counts of label '0': {} \n".format(sum(y_train == 0 )))

from imblearn.under_sampling import NearMiss
nr = NearMiss()

X_train_miss, y_train_miss = (X_train, y_train.ravel())

print('After Undersampling, the shape of train_X: {}'.format(X_train_miss.shape))
print('After Undersampling, the shape of train_X: {} \n'.format(y_train_miss.shape))

print("After Undersampling, counts of label '1': {}".format(sum(y_train_miss == 1)))
print("After Undersampling, counts of label '0': {}".format(sum(y_train_miss == 0)))
```

```
lr2 = LogisticRegression()
lr2.fit(X_train_miss, y_train_miss.ravel())
predictions = lr2.predict(X_test)

print(classification_report(y_test, prediction))
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-1-48a4cba32ab8> in <cell line: 1>()
----> 1 lr2 = LogisticRegression()
      2 lr2.fit(X_train_miss, y_train_miss.ravel())
      3 predictions = lr2.predict(X_test)
      4 
      5 print(classification_report(y_test, prediction))

NameError: name 'LogisticRegression' is not defined
```