import libraries

In [1]:
```
#Importing the basic libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import plotly.offline as py
import seaborn as sns
from plotly import tools
py.init_notebook_mode(connected=True)
import plotly.graph_objs as go
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, f1_score, precision_score, recall_score, classification_report, confusion_matrix
import warnings
warnings.filterwarnings('ignore')
```

import dataset

In [2]:
```
#Reading the dataset
offers = pd.read_csv('H:\Level 4 Information Systems\Plastikat\Plastikat Data\offers_Classification.csv',
                     encoding=('ISO-8859-1'), low_memory=False)
```

Define X and Y

In [3]:
```
#Dividing the dataframe into x features and y target variable
x = offers.iloc[:, :-1]
y = offers.iloc[:, 5]


x.columns = ['user_exceed_min_quantity', 'plastic_logical_in_prev_trans',
'user_exist_in_prev_trans', 'user_cheating', 'reported_by_delegate']
y.columns=['offer_decision']


x.head()
```

Out[3]:

	user_exceed_min_quantity	plastic_logical_in_prev_trans	user_exist_in_prev_trans	user_cheating	reported_by_delegate
0	No	Yes	No	No	Yes
1	Yes	Yes	Yes	No	Yes
2	No	Yes	No	No	Yes
3	No	No	No	No	No
4	Yes	No	No	Yes	No

encode the data

In [4]:
```
#Using pandas dummies function to encode the data into categorical data
x = pd.get_dummies(x, prefix_sep='_', drop_first=True)
x.head(-5)
```

Out[4]:

	user_exceed_min_quantity_Yes	plastic_logical_in_prev_trans_Yes	user_exist_in_prev_trans_Yes	user_cheating_Yes	reported_by_delegate_Yes
0	0	1	0	0	1
1	1	1	1	0	1
2	0	1	0	0	1
3	0	0	0	0	0
4	1	0	0	1	0
...	...	...	...	...	...
1307	1	1	1	0	0
1308	1	1	1	0	1
1309	0	0	1	0	1
1310	0	1	0	0	1
1311	1	0	1	1	0

1312 rows × 5 columns

split the data into teaining and testing data

In [5]:
```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.25, random_state = 0)
```

logistic regression

In [6]:
```
#Using logistic regression
from sklearn.linear_model import LogisticRegression
clf = LogisticRegression(random_state = 0)
clf.fit(x_train, y_train)

y_pred = clf.predict(x_test)
f1_LR=f1_score(y_test,y_pred, average='macro')
print("Training Accuracy: ",clf.score(x_train, y_train))
print("Testing Accuracy: ", clf.score(x_test, y_test))
cm = confusion_matrix(y_test, y_pred)
print(cm)
print(classification_report(y_test,y_pred))
```

```
Training Accuracy:  0.9594731509625126
Testing Accuracy:  0.9696969696969697
[[138   0]
 [ 10 182]]
              precision    recall  f1-score   support

      accept       0.93      1.00      0.97       138
      reject       1.00      0.95      0.97       192

    accuracy                           0.97       330
   macro avg       0.97      0.97      0.97       330
weighted avg       0.97      0.97      0.97       330
```

naive_bayes classifier

In [7]:
```
#Using NB classifier
from sklearn.naive_bayes import GaussianNB
clf = GaussianNB()
clf.fit(x_train, y_train)


y_pred = clf.predict(x_test)
f1_NB=f1_score(y_test,y_pred, average='macro')
print("Training Accuracy: ",clf.score(x_train, y_train))
print("Testing Accuracy: ", clf.score(x_test, y_test))
cm = confusion_matrix(y_test, y_pred)
print(cm)
print(classification_report(y_test,y_pred))
```

```
Training Accuracy:  0.9098277608915907
Testing Accuracy:  0.8939393939393939
[[113  25]
 [ 10 182]]
              precision    recall  f1-score   support

      accept       0.92      0.82      0.87       138
      reject       0.88      0.95      0.91       192

    accuracy                           0.89       330
   macro avg       0.90      0.88      0.89       330
weighted avg       0.90      0.89      0.89       330
```

K-nearest Neighbor classifier

In [8]:
```
#Using KNN classifier
from sklearn.neighbors import KNeighborsClassifier
clf = KNeighborsClassifier(n_neighbors = 5, metric = 'minkowski', p = 2)
clf.fit(x_train, y_train)

y_pred = clf.predict(x_test)
f1_KNN=f1_score(y_test,y_pred, average='macro')
print("Training Accuracy: ",clf.score(x_train, y_train))
print("Testing Accuracy: ", clf.score(x_test, y_test))
cm = confusion_matrix(y_test, y_pred)
print(cm)
print(classification_report(y_test,y_pred))
```

```
Training Accuracy:  1.0
Testing Accuracy:  1.0
[[138   0]
 [  0 192]]
              precision    recall  f1-score   support

      accept       1.00      1.00      1.00       138
      reject       1.00      1.00      1.00       192

    accuracy                           1.00       330
   macro avg       1.00      1.00      1.00       330
weighted avg       1.00      1.00      1.00       330
```

Decision Tree Classifier

In [9]:
```
#Trying decision tree classifier
from sklearn.tree import DecisionTreeClassifier
clf = DecisionTreeClassifier(criterion = 'entropy', random_state = 0)
clf.fit(x_train, y_train)

y_pred = clf.predict(x_test)
f1_DT=f1_score(y_test,y_pred, average='macro')
print("Training Accuracy: ",clf.score(x_train, y_train))
print("Testing Accuracy: ", clf.score(x_test, y_test))
cm = confusion_matrix(y_test, y_pred)
print(cm)
print(classification_report(y_test,y_pred))
```

```
Training Accuracy:  1.0
Testing Accuracy:  1.0
[[138   0]
 [  0 192]]
              precision    recall  f1-score   support

      accept       1.00      1.00      1.00       138
      reject       1.00      1.00      1.00       192

    accuracy                           1.00       330
   macro avg       1.00      1.00      1.00       330
weighted avg       1.00      1.00      1.00       330
```

Comaprison between algorithms score

In [10]:
```
models=['Naive Bayes Classifier','Logistic Regression','K-nearest Neighbour','Decision Tree Classifier']
fig = go.Figure(data=[
    go.Bar(name='f1_score', x=models, y=[f1_NB,f1_LR,f1_KNN,f1_DT])])
fig.show()
```