```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
dataset = pd.read_csv("/content/linkedin_data.csv", encoding = 'latin-1')
```

```
dataset.head(3)
```

	Unnamed: 0	avg_n_pos_per_prev_tenure	avg_pos_len	avg_prev_tenure_len	c_name	
0	0	2.000000	457.0	1338.0	TD	ur
1	1	1.500000	212.0	897.5	Light Up The World (LUTW)	ur
2	2	1.333333	243.0	669.0	Glacier	ur

3 rows × 52 columns

```
dataset.shape
```

```
(62709, 52)
```

```
dataset.info()
```

```
RangeIndex: 62709 entries, 0 to 62708
Data columns (total 52 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   Unnamed: 0                 62709 non-null  int64
 1   avg_n_pos_per_prev_tenure  62709 non-null  float64
 2   avg_pos_len                62709 non-null  float64
 3   avg_prev_tenure_len        62709 non-null  float64
 4   c_name                     62703 non-null  object
 5   m_urn                      62709 non-null  object
 6   n_pos                      62709 non-null  int64
 7   n_prev_tenures             62709 non-null  int64
 8   tenure_len                 62709 non-null  int64
 9   age                        62709 non-null  int64
 10  beauty                     62709 non-null  float64
 11  beauty_1                   62709 non-null  float64
 12  beauty_0                   62709 non-null  float64
 13  blur                       62709 non-null  float64

 14  blur_gaussian              62709 non-null  float64
 15  blur_motion                62709 non-null  float64
```

```
 16  emo_anger                 62709 non-null  float64
 17  emo_disgust               62709 non-null  float64
 18  emo_fear                  62709 non-null  float64
 19  emo_happiness             62709 non-null  float64
 20  emo_neutral               62709 non-null  float64
 21  emo_sadness               62709 non-null  float64
 22  emo_surprise              62709 non-null  float64
 23  ethnicity                 62709 non-null  int64
 24  face_quality              62709 non-null  float64
 25  gender                    62709 non-null  int64
 26  glass                     62709 non-null  int64
 27  head_pitch                62709 non-null  float64
 28  head_roll                 62709 non-null  float64
 29  head_yaw                  62709 non-null  float64
 30  img                       62709 non-null  object
 31  mouth_close               62709 non-null  float64
 32  mouth_mask                62709 non-null  float64
 33  mouth_open                62709 non-null  float64
 34  mouth_other               62709 non-null  float64
 35  skin_acne                 62709 non-null  float64
 36  skin_2_circle             62709 non-null  float64
 37  skin_health               62709 non-null  float64
 38  skin_stain                62709 non-null  float64
 39  smile                     62709 non-null  float64
 40  african                   62709 non-null  float64
 41  celtic_english            62709 non-null  float64
 42  east_0                    62709 non-null  float64
 43  european                  62709 non-null  float64
 44  greek                     62709 non-null  float64
 45  hispanic                  62709 non-null  float64
 46  jewish                    62709 non-null  float64
 47  muslim                    62709 non-null  float64
 48  nationality               62709 non-null  object
 49  nordic                    62709 non-null  float64
 50  south_0                   62709 non-null  float64
 51  n_followers               62709 non-null  int64
dtypes: float64(39), int64(9), object(4)
memory usage: 24.9+ MB
```

```
dataset.corr()
```

	Unnamed: 0	avg_n_pos_per_prev_tenure	avg_pos_len	avg_pr
Unnamed: 0	1.000000	0.006258	-0.000785	
avg_n_pos_per_prev_tenure	0.006258	1.000000	-0.051867	
avg_pos_len	-0.000785	-0.051867	1.000000	
avg_prev_tenure_len	0.005987	0.416482	0.227649	
n_pos	0.000996	0.012144	0.018707	
n_prev_tenures	-0.000368	-0.049366	-0.146736	
tenure_len	0.000293	-0.039973	0.772553	
age	-0.002605	0.038474	0.150765	
beauty	-0.008886	-0.025831	-0.111981	
beauty_1	-0.009072	-0.020071	-0.093928	
beauty_0	-0.008593	-0.027889	-0.109249	
blur	-0.000776	-0.019832	0.002119	
blur_gaussian	-0.000776	-0.019832	0.002119	
blur_motion	-0.000776	-0.019832	0.002119	
emo_anger	-0.004784	-0.008238	0.012294	
emo_disgust	-0.008281	-0.009413	0.005171	
emo_fear	-0.013878	-0.007312	0.002076	
emo_happiness	-0.002485	0.032011	0.012085	
emo_neutral	0.017022	-0.026270	-0.016148	
emo_sadness	-0.010713	-0.007598	-0.003627	
emo_surprise	-0.020064	-0.004097	-0.007044	
ethnicity	-0.002900	0.003034	0.014183	
face_quality	0.008575	0.021930	0.015202	
gender	0.002340	0.001375	-0.053958	
glass	0.000566	0.004270	0.017719	
head_pitch	-0.008349	-0.001790	-0.028070	
head_roll	0.000962	0.006641	-0.005335	
head_yaw	0.010728	-0.009362	-0.001080	
mouth_close	0.016739	-0.026987	-0.001947	
mouth_mask	-0.008880	-0.005893	0.015996	
mouth_open	-0.014760	0.027101	-0.005041	
mouth_other	0.000592	0.000621	0.008544	

skin_acne	-0.002912	-0.000110	0.000238
skin_2_circle	-0.004510	0.001670	0.014150
skin_health	0.000064	-0.013209	-0.062376
skin_stain	-0.005812	0.020001	0.041752
smile	-0.007662	0.029545	0.004145
african	-0.006654	0.002040	0.003256
celtic_english	-0.002102	0.031945	0.066172
east_0	0.019499	-0.002803	-0.027242
european	-0.014338	-0.000545	-0.000199
greek	0.005837	0.007884	-0.006207

```
dataset.describe()
```

	Unnamed: 0	avg_n_pos_per_prev_tenure	avg_pos_len	avg_prev_tenure_len	
count	62709.000000	62709.000000	62709.000000	62709.000000	6.
mean	31354.000000	1.194319	765.657189	1100.783854	
std	18102.673352	0.506714	750.725210	985.744936	
min	0.000000	1.000000	-120.000000	0.000000	
25%	15677.000000	1.000000	274.000000	537.000000	
50%	31354.000000	1.000000	578.000000	882.833333	
75%	47031.000000	1.200000	1035.000000	1399.833333	
max	62708.000000	15.000000	21884.000000	39781.000000	

8 rows × 48 columns

```
dataset.drop('c_name', axis='columns', inplace=True)
dataset.drop('m_urn', axis='columns', inplace=True)
```

```
dataset.drop('img', axis='columns', inplace=True)
```

```
dataset.drop('nationality', axis='columns', inplace=True)
# dataset.drop('gender', axis='columns', inplace = True)
```

```
x = dataset.drop('gender',axis='columns')
y = dataset['gender']
```

```
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, random_state = 0)
```

```
# dataset.drop('glass', axis = "columns", inplace = True)
```

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.fit_transform(x_test)
```

K Nearest Neighbors

```
from sklearn.neighbors import KNeighborsClassifier
model = KNeighborsClassifier(n_neighbors=3, metric = 'minkowski', p=2)
model.fit(x_train, y_train)
```

```
KNeighborsClassifier(n_neighbors=3)
```

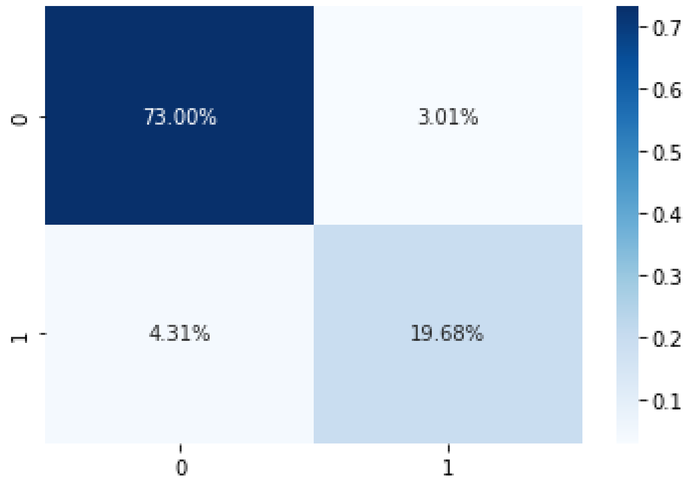
```
y_pred = model.predict(x_test)
```

```
from sklearn.metrics import accuracy_score, confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print(cm)
accuracy_score(y_test, y_pred)
```

```
[[13734   566]
 [  811  3702]]
0.9268059320682507
```

```
sns.heatmap(cm/np.sum(cm), annot=True,fmt='.2%', cmap='Blues')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8592d13290>
```




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

```
              precision    recall  f1-score   support

           0       0.94      0.96      0.95     14300
           1       0.87      0.82      0.84      4513

    accuracy                           0.93     18813
   macro avg       0.91      0.89      0.90     18813
weighted avg       0.93      0.93      0.93     18813
```

Random Forest Regressor

```
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier()
rf.fit(x_train , y_train)
```

```
RandomForestClassifier()
```

```
y_pred1 = rf.predict(x_test)
```

```
from sklearn.metrics import accuracy_score
acc= accuracy_score(y_test,y_pred1)
acc
```

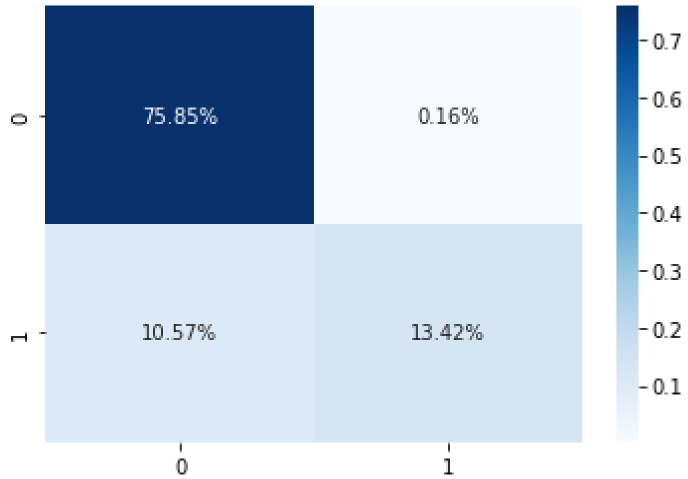
```
0.8926274384733961
```

```
from sklearn.metrics import confusion_matrix, precision_score, accuracy_score, recall_scor
cm = confusion_matrix(y_test.values , y_pred1)
cm
```

```
array([[14269,    31],
       [ 1989,  2524]])
```

```
sns.heatmap(cm/np.sum(cm), annot=True,fmt='.2%', cmap='Blues')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f859a3e3110>
```



```
print(metrics.classification_report(y_test,y_pred1))
```

```
              precision    recall  f1-score   support

           0       0.88      1.00      0.93     14300
           1       0.99      0.56      0.71      4513

    accuracy                           0.89     18813
   macro avg       0.93      0.78      0.82     18813
weighted avg       0.90      0.89      0.88     18813
```

Logistic Regression

```
from sklearn.linear_model import LogisticRegression
model3 = LogisticRegression()
model3.fit(x_train, y_train)
```

```
LogisticRegression()
```

```
y_pred5=model3.predict(x_test)
```

```
from sklearn.metrics import accuracy_score
accuracy_score(y_test,y_pred5)
```

```
0.8786477435815659
```

```
print(metrics.classification_report(y_test,y_pred5))
```

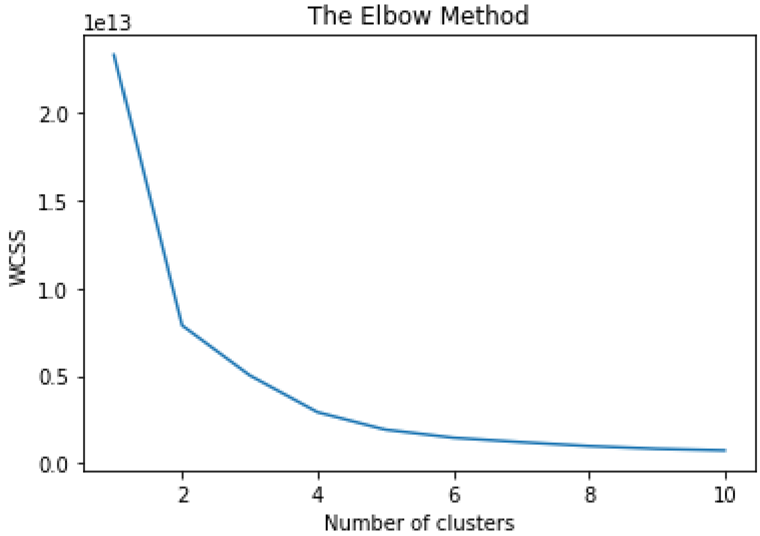
```
              precision    recall  f1-score   support

           0       0.90      0.95      0.92     14300
           1       0.81      0.65      0.72      4513

    accuracy                           0.88     18813
   macro avg       0.85      0.80      0.82     18813
weighted avg       0.87      0.88      0.87     18813
```

K - Means Clustering

```
from sklearn.cluster import KMeans
```

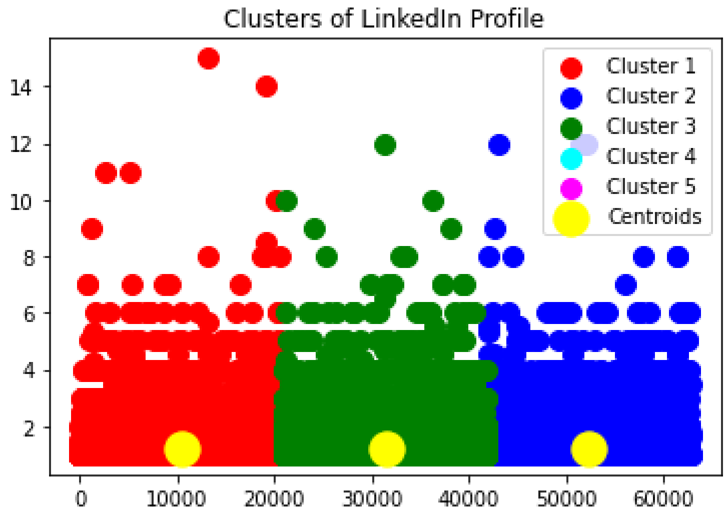
```
from sklearn.cluster import KMeans
wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters = i, init = 'k-means++', random_state = 42)
    kmeans.fit(x)
```

```
    wcss.append(kmeans.inertia_)
plt.plot(range(1, 11), wcss)
plt.title('The Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
```




```
kmeans = KMeans(n_clusters = 3, init = 'k-means++', random_state = 42)
y_kmeans = kmeans.fit_predict(x)
x = np.array(x)
```

```
plt.scatter(x[y_kmeans == 0, 0], x[y_kmeans == 0, 1], s = 100, c = 'red', label = 'Cluster
plt.scatter(x[y_kmeans == 1, 0], x[y_kmeans == 1, 1], s = 100, c = 'blue', label = 'Cluste
plt.scatter(x[y_kmeans == 2, 0], x[y_kmeans == 2, 1], s = 100, c = 'green', label = 'Clust
plt.scatter(x[y_kmeans == 3, 0], x[y_kmeans == 3, 1], s = 100, c = 'cyan', label = 'Cluste
plt.scatter(x[y_kmeans == 4, 0], x[y_kmeans == 4, 1], s = 100, c = 'magenta', label = 'Clu
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s = 300, c = 'ye
plt.title('Clusters of LinkedIn Profile')
plt.legend()
plt.show()
```



Support Vector Machine

```
from sklearn.svm import SVC
classifier1 = SVC(kernel = 'linear', random_state = 0)
classifier1.fit(x_train, y_train)
```

```
SVC(kernel='linear', random_state=0)
```

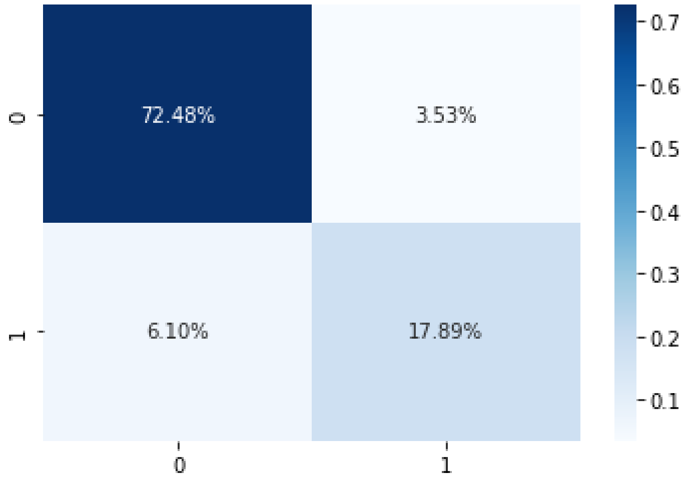
```
y_pred4 = classifier1.predict(x_test)
```

```
cm = confusion_matrix(y_test, y_pred)
print(cm)
accuracy_score(y_test, y_pred4)
```

```
[[13636   664]
 [ 1148  3365]]
0.9050656460957849
```

```
sns.heatmap(cm/np.sum(cm), annot=True,fmt='.2%', cmap='Blues')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8596b6c7d0>
```




```
print(metrics.classification_report(y_test,y_pred4))
```

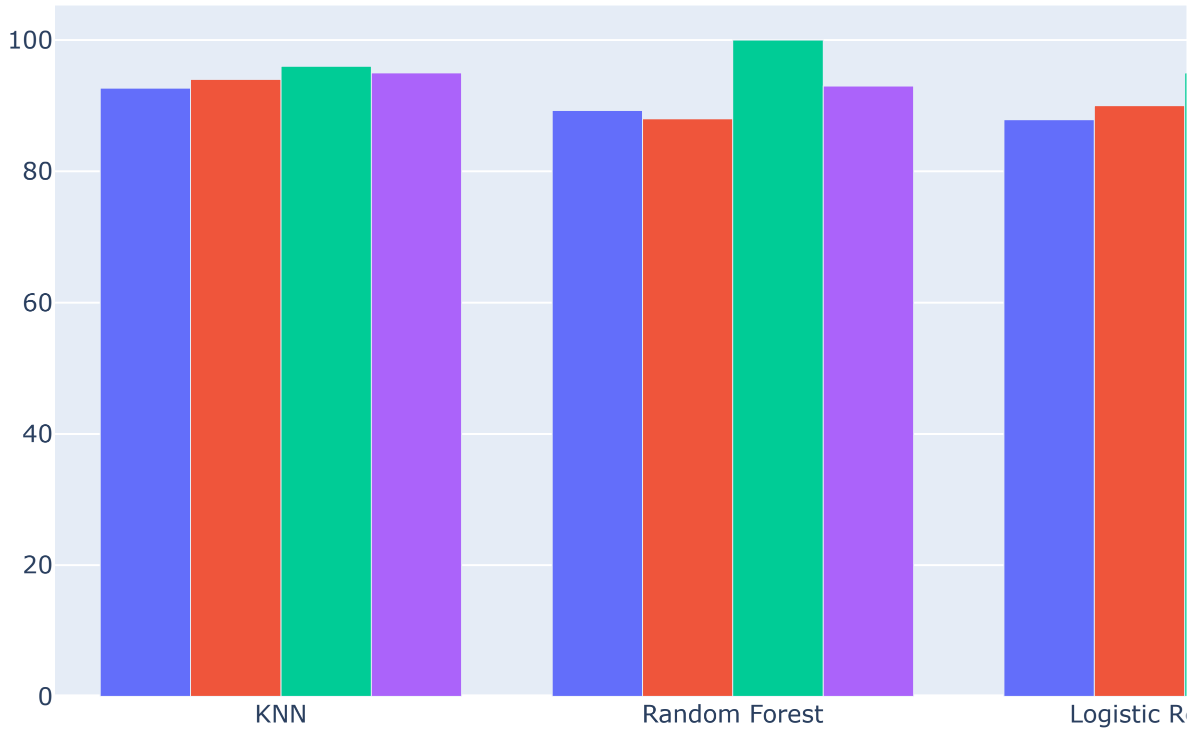
```
              precision    recall  f1-score   support

           0       0.89      1.00      0.94     14300
           1       0.99      0.61      0.76      4513

    accuracy                           0.91     18813
   macro avg       0.94      0.80      0.85     18813
weighted avg       0.91      0.91      0.90     18813
```

Bar Graph

```
pip install chart-studio
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/
Collecting chart-studio
  Downloading chart_studio-1.1.0-py3-none-any.whl (64 kB)
     |████████████████████████████████| 64 kB 2.5 MB/s
Collecting retrying>=1.3.3
  Downloading retrying-1.3.4-py3-none-any.whl (11 kB)
Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-packages (f
Requirement already satisfied: plotly in /usr/local/lib/python3.7/dist-packages (fro
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (from c
Requirement already satisfied: tenacity>=6.2.0 in /usr/local/lib/python3.7/dist-pack
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist-p
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-pa
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-package
Installing collected packages: retrying, chart-studio
Successfully installed chart-studio-1.1.0 retrying-1.3.4
```

```
from plotly import *
import chart_studio.plotly as pyt
from plotly.graph_objs import *
import plotly.graph_objs as go
from  plotly.offline import iplot, init_notebook_mode

trace1 = {
  "name": "Accuracy",
  "type": "bar",
  "x": ["KNN", "Random Forest", "Logistic Regression", "SVM"],
  "y": [92.68, 89.26, 87.86, 90.50]
}
trace2 = {
  "name": "Precision",
  "type": "bar",
  "x": ["KNN", "Random Forest", "Logistic Regression", "SVM"],
  "y": [94, 88, 90, 89]
}
trace3 = {
  "name": "Recall",
  "type": "bar",
  "x": ["KNN", "Random Forest", "Logistic Regression", "SVM"],
  "y": [96, 100, 95, 100]
}
trace4 = {
  "name": "F1 Score",
  "type": "bar",
  "x": ["KNN", "Random Forest", "Logistic Regression", "SVM"],
  "y": [95, 93, 92, 94]
}
data = Data([trace1, trace2, trace3, trace4])
layout = {"barmode": "group"}
fig = Figure(data=data, layout=layout)
plot_url = iplot(fig)
```



0s completed at 7:50 PM