SHRI VILE PARLE KELAVANI MANDAL'S

SHRI BHAGUBHAI MAFATLALPOLYTECHNIC

PROGRAM: INFORMATION TECHNOLOGY

COURSE NAME: PYTHON PROGRAMMING (ML)

COURSE CODE: MLP190910

SEMESTER: V

TITLE: LINKEDIN RECOMMENDATION SYSTEM USING MACHINE LEARNING

FACULTY: MRS. SWAPNA NAIK

STUDENT NAME 1: VATSAL PRASHANT KOTHA

STUDENT ROLL NO 1: T117

STUDENT NAME 2: JAINAM JAYKUMAR SHETH

STUDENT ROLL NO 2: T122

ABSTRACT

Online professional social networks like LinkedIn are crucial in assisting both employers and job seekers in connecting with qualified prospects. The purpose of LinkedIn's employment ecosystem is to provide tools for connecting job seekers and employers, as well as to act as a marketplace for effective matching between prospective candidates and job posts. A key tool for achieving these objectives is LinkedIn's job suggestions product, which presents members with individualized lists of suggested job postings based on the structured, context-specific information in their profiles.

The LinkedIn Recruiter product provides a ranked list of candidates corresponding to a search request in the form of a query, a job posting, or a recommended candidate. Given a search request, candidates that match the request are selected and then ranked based on a variety of factors (such as the similarity of their age, gender, work experience/skills with the search criteria, job posting location, and the likelihood of a response from an interested candidate) using machine-learned models in multiple passes.

SOFTWARE REQUIREMENTS

1. PYTHON

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. It was created by Guido van Rossum during 1985-1990. Like Perl, Python source code is also available under the GNU General Public License (GPL). This tutorial gives enough understanding on Python programming language.

2. GOOGLE COLABORATORY

Colaboratory, or “Colab” for short, is a product from Google Research. Colab allows anybody to write and execute arbitrary python code through the browser, and is especially well suited to machine learning, data analysis and education. More technically, Colab is a hosted Jupyter notebook service that requires no setup to use, while providing access free of charge to computing resources including GPUs.

HARDWARE REQUIREMENTS

1. **WINDOWS 10**
2. **8TH GEN INTEL CORE I5 PROCESSOR 1.80 GHZ**
3. **8.00 GB RAM**
4. **350 GB FREE DISK SPACE**

PROJECT SOURCE CODE

```
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)
dataset.shape
```

```
(62709, 52)
```

```
dataset.info()
dataset.corr()
dataset.describe()
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)
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)
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.fit_transform(x_test)
```

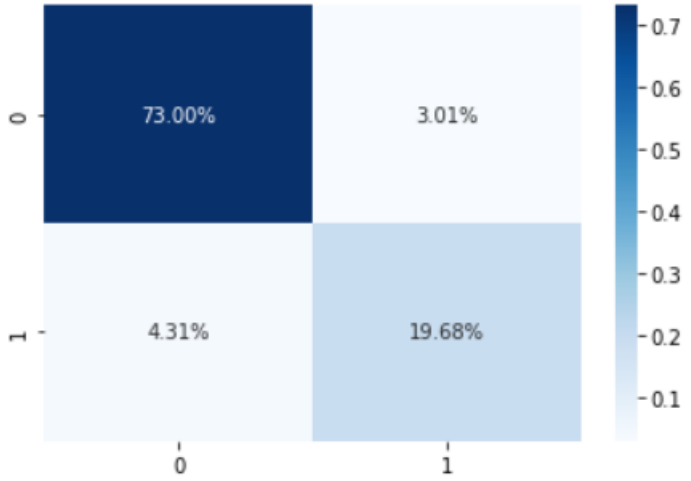
K – NEAREST NEIGHBOR

```
from sklearn.neighbors import KNeighborsClassifier
model = KNeighborsClassifier(n_neighbors=3, metric = 'minkowski', p=2)
model.fit(x_train, y_train)
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)
y_pred1 = rf.predict(x_test)
from sklearn.metrics import accuracy_score
acc= accuracy_score(y_test,y_pred1)
acc
```

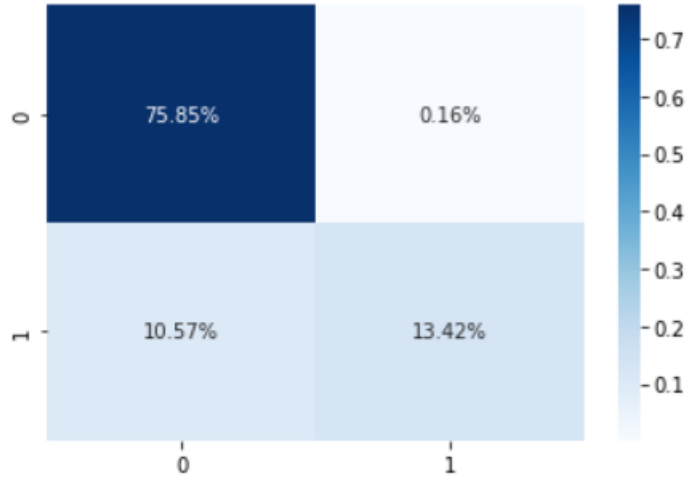
```
0.8926274384733961
```

```
from sklearn.metrics import confusion_matrix, precision_score, accuracy
_score, recall_score
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)
y_pred5=model3.predict(x_test)
from sklearn.metrics import accuracy_score
accuracy_score(y_test,y_pred5)
```

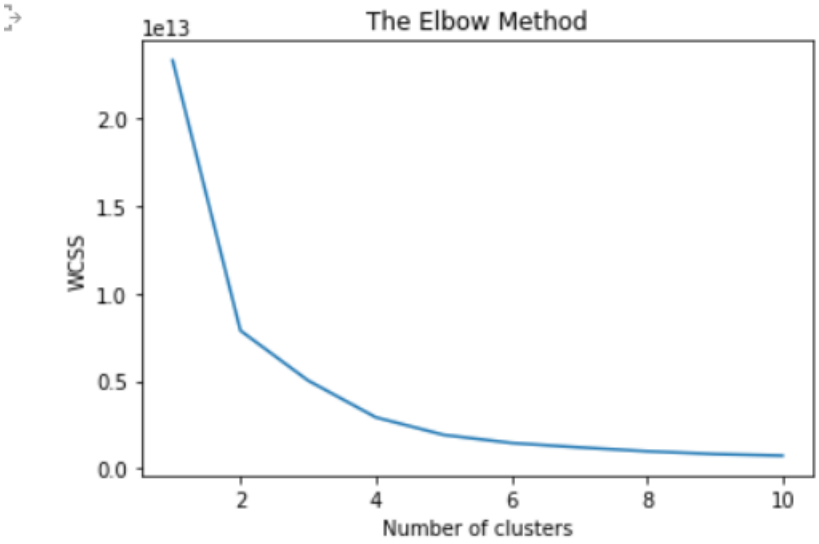
```
0.8786477435815659
```

```
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

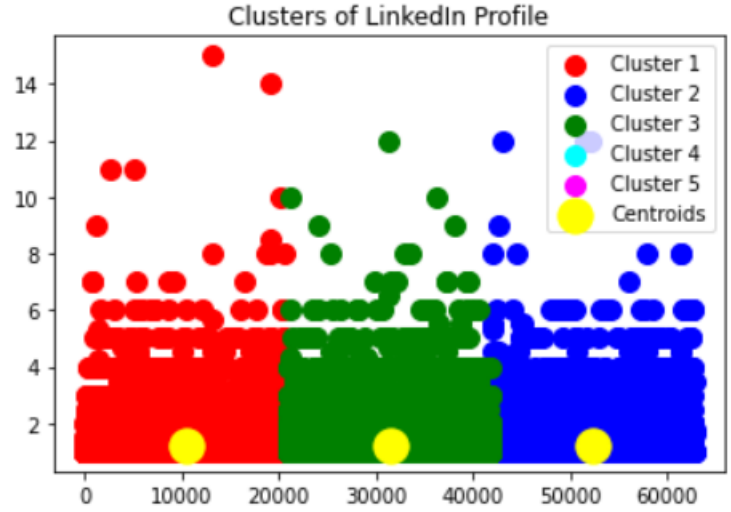
K-MEANS CLUSTERING

```
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 1')
plt.scatter(x[y_kmeans == 1, 0], x[y_kmeans == 1, 1], s = 100, c = 'blu
e', label = 'Cluster 2')
plt.scatter(x[y_kmeans == 2, 0], x[y_kmeans == 2, 1], s = 100, c = 'gre
en', label = 'Cluster 3')
plt.scatter(x[y_kmeans == 3, 0], x[y_kmeans == 3, 1], s = 100, c = 'cya
n', label = 'Cluster 4')
plt.scatter(x[y_kmeans == 4, 0], x[y_kmeans == 4, 1], s = 100, c = 'mag
enta', label = 'Cluster 5')
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1
], s = 300, c = 'yellow', label = 'Centroids')
plt.title('Clusters of LinkedIn Profile')
plt.legend()
```

```
plt.show()
```



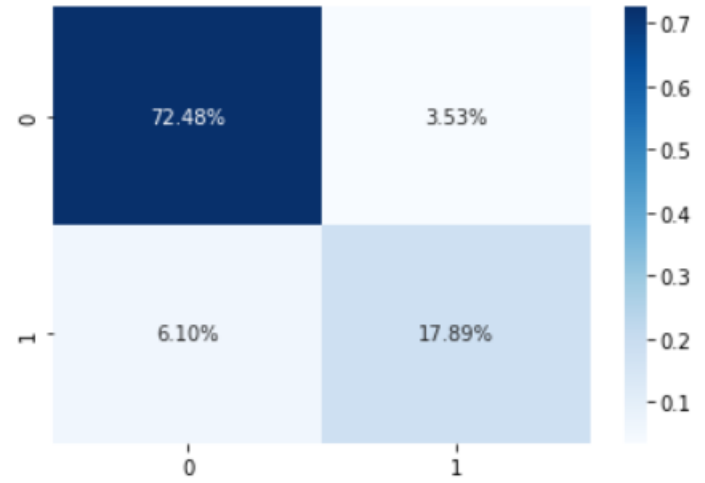
SUPPORT VECTOR MACHINE

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



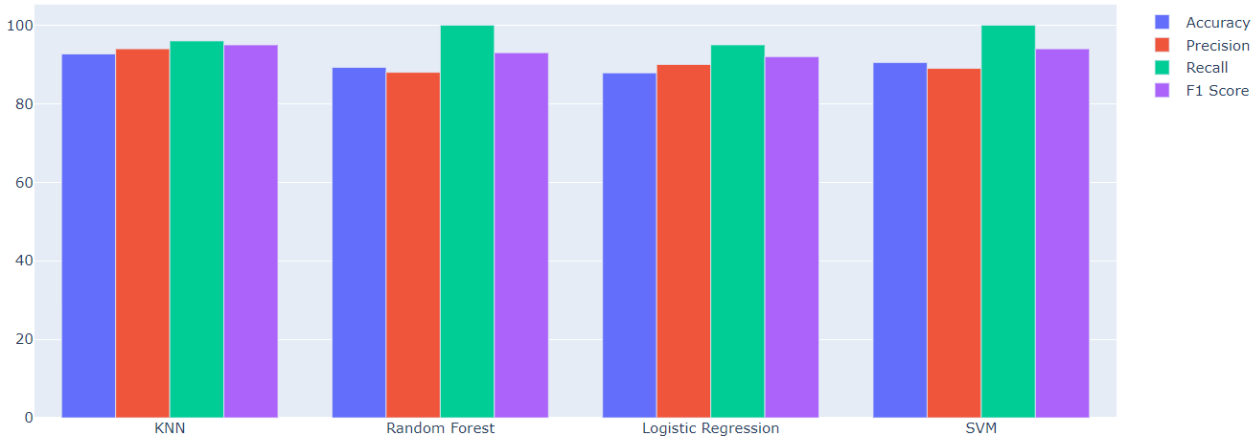

```
print(metrics.classification_report(y_test,y_pred4))
```

	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

PLOTTING BAR GRAPHS OF ALL MODELS

```
pip install chart-studio
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)
```


100
80
60
40
20
0
KNN
Random Forest
Logistic Regression
SVM
Accuracy
Precision
Recall
F1 Score

CONCLUSION

The Recommendation System of LinkedIn uses machine learning techniques including KNearestNeighbor, Random Forest Regressor, Logistic Regression, K-Means Clustering and Support Vector Machine. 92.68% accuracy was obtained by using KNearestNeighbor Machine Learning technique. From the results of our analysis, we infer that the KNearestNeighbor is the machine learning algorithm which predicted the best accuracy in comparison with other machine learning models. To conclude, a bar including accuracy, f1-score, recall, and precision was plotted. With the help of this project, recommendations will be much easier, precise and accurate.