PROJECT REPORT

CAMPUS RECRUITMENT USING MACHINE LEARNING

By- Mr. Gaurav Kumar

Email: gauravkr2508@gmail.com

Phone: 9359271595

1.INTRODUCTION

Placement of students is one of the most important objectives of an educational institution. Reputation and yearly admission of an institution is directly depends upon the placements it provides to students. Every student take admission to the college by seeing the percentage of placements in the college. That is why all the institution tries to strengthen their placement department. So seeing the need of the time, this algorithm is all about predicting whether the student will be placed or not in Campus Recruitment. For this algorithm, student's previous year record has been taken into the account

2.PROBLEM STATEMENT

Every Machine Learning project starts by understanding the problem as well as understanding the data available in hand. In this Dataset there are total 15 features (Age, HSC marks, SSC marks, Work Experience, Specialisation, etc) and 215 Data Points. On this basis we have tried to find the status whether he will get placed or not through the campus. The algorithms included K Neighbors Classifier, Support Vector Classifier, Decision Tree Classifier and Random Forest Classifier. Logistic Regression

3.METHODOLOGY

The whole approach is depicted by the following flowchart (Figure 1).

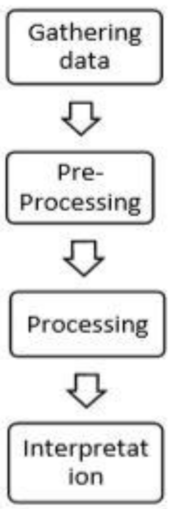


Figure 1

3.1 Data Gathering:

The sample data has been given by project manager which was in .csv format consisting of all the previous records of a student consisting Age, HSC marks, SSC marks, Work Experience, Specialisation, etc. The data collected consists of over 200 instances of students.

Imported Libraries:

```
import os      # 'os' library to change the woking directory
import pandas as pd  #pandas library to work with dataframe
import numpy as np   # numpy libraries to work with arrays
import seaborn as sns  #seaborn libraries for statistical data visualisation
import matplotlib.pyplot as plt  #for creating static, animated, and interactive visualizations.
```

I downloaded .csv file and named it as datasets_Placement_Data1 and stored it under df by importing ***pandas library.***

3.2 Pre-Processing:

Data pre-processing is a technique that is used to convert raw data into a clean dataset. The data is gathered from different sources is always in a raw format which is not feasible for the analysis.

Pre-processing process consist of following methods:

3.2.1] Check the null values from the dataset:

Missing Data can occur when no information is provided for one or more items or for a whole unit. Missing Data is a very big problem in real life scenario. It is sometimes regarded as NaN (Not a Number) in data analysis. Here we calculated the total no of null values by using *isna().sum() function.*

```
os.chdir(r"C:\Users\GK\Downloads")
df=pd.read_csv("datasets_Placement_Data1.csv",index_col=0)
df.isna().sum()
```

```
gender            0
ssc_p             0
ssc_b             0
hsc_p             0
hsc_b             0
hsc_s             0
degree_p          0
degree_t          0
workex            0
etest_p           0
specialisation    0
mba_p             0
status            0
salary           67
dtype: int64
```

3.2.2] **Fill the null values using median or mode:**

There are lot of ways to impute the gaps and in most cases we take a help of Mean, Median, mode or sometime variance. After analysing data, salary of every null values of not_placed students, kept as Zero. Because they are unemployed. After this I printed first 5 data using df.head() function shown below.

```
df["salary"]=df["salary"].fillna(df["salary"].fillna(0)) #filled every null values of salary with Zero because they are unemploy
df.head()
```

sl_no	gender	ssc_p	ssc_b	hsc_p	hsc_b	hsc_s	degree_p	degree_t	workex	etest_p	specialisation	mba_p	status	salary
1	M	67.00	Others	91.00	Others	Commerce	58.00	Sci&Tech	No	55.0	Mkt&HR	58.80	Placed	270000.0
2	M	79.33	Central	78.33	Others	Science	77.48	Sci&Tech	Yes	86.5	Mkt&Fin	66.28	Placed	200000.0
3	M	65.00	Central	68.00	Central	Arts	64.00	Comm&Mgmt	No	75.0	Mkt&Fin	57.80	Placed	250000.0
4	M	56.00	Central	52.00	Central	Science	52.00	Sci&Tech	No	66.0	Mkt&HR	59.43	Not Placed	0.0
5	M	85.80	Central	73.60	Central	Commerce	73.30	Comm&Mgmt	No	96.8	Mkt&Fin	55.50	Placed	425000.0

As you can see in *salary* coloumn that every NaN is replaced by zero.

3.2.3] **Remove the insignificant columns from the dataset if necessary.**

When we get any dataset, not necessarily every column (feature) is going to have an impact on the output variable. If we add these irrelevant features in the model, it will just make the model worst. To avoid this situation we removed Sr. no. of student because it will not make an impact on student's placement. It can be done by using *index_col=0* while reading .csv file .After that I used df.describe, The **describe() function** computes a summary of statistics pertaining to the Data Frame columns. This **function** gives the mean, std and IQR values.

3.2.4] **Convert the categorical columns to numerical columns.**

In machine learning projects, one important part is feature engineering. It is very common to see categorical features in a dataset. However, machine learning algorithm can only read numerical values. It is essential to encoding categorical features into numerical values. To make our task easier, *Scikit-learn* library gives *us LabelEncoder* in whch we can convert catergorical data into numerical data.

sl_no	gender	ssc_p	ssc_b	hsc_p	hsc_b	hsc_s	degree_p	degree_t	workex	etest_p	specialisation	mba_p	status	salary
1	1	67.00	1	91.00	1	1	58.00	2	0	55.0	1	58.80	1	270000.0
2	1	79.33	0	78.33	1	2	77.48	2	1	86.5	0	66.28	1	200000.0
3	1	65.00	0	68.00	0	0	64.00	0	0	75.0	0	57.80	1	250000.0
4	1	56.00	0	52.00	0	2	52.00	2	0	66.0	1	59.43	0	0.0
5	1	85.80	0	73.60	0	1	73.30	0	0	96.8	0	55.50	1	425000.0

As you can see in this fig that every categorical column(gender,workex.specialisation,status) has been now converted into numerical data.

3.2.5] **Find the statistical information of the dataset.**

Sometimes, when facing a Data problem, we must first dive into the Dataset and learn about it. Its properties, its distributions — we need to immerse in the domain. Statistical information of dataset consists of mean, median, mode, variance, correlation, etc. This can be done by *.info() & .describe() function*. Using this, we can get statistical information related to our data file.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 215 entries, 1 to 215
Data columns (total 14 columns):
gender            215 non-null int64
ssc_p             215 non-null float64
ssc_b             215 non-null int64
hsc_p             215 non-null float64
hsc_b             215 non-null int64
hsc_s             215 non-null int64
degree_p          215 non-null float64
degree_t          215 non-null int64
workex            215 non-null int64
etest_p           215 non-null float64
specialisation    215 non-null int64
mba_p             215 non-null float64
status            215 non-null int64
salary            215 non-null float64
dtypes: float64(6), int64(8)
memory usage: 35.2 KB
```

In this fig we can see that no column has null value .

```
df.describe()
```

	ssc_p	hsc_p	degree_p	etest_p	mba_p	salary
count	215.000000	215.000000	215.000000	215.000000	215.000000	148.000000
mean	67.303395	66.333163	66.370186	72.100558	62.278186	288655.405405
std	10.827205	10.897509	7.358743	13.275956	5.833385	93457.452420
min	40.890000	37.000000	50.000000	50.000000	51.210000	200000.000000
25%	60.600000	60.900000	61.000000	60.000000	57.945000	240000.000000
50%	67.000000	65.000000	66.000000	71.000000	62.000000	265000.000000
75%	75.700000	73.000000	72.000000	83.500000	66.255000	300000.000000
max	89.400000	97.700000	91.000000	98.000000	77.890000	940000.000000

In this fig we can see various statistical information such as mean, median, mode, max etc.

4. Confusion Matrix:

In the field of machine learning and specifically the problem of statistical classification, a confusion matrix, also known as an error matrix, is a specific table layout that allows visualization of the performance of an algorithm, typically a supervised learning one.

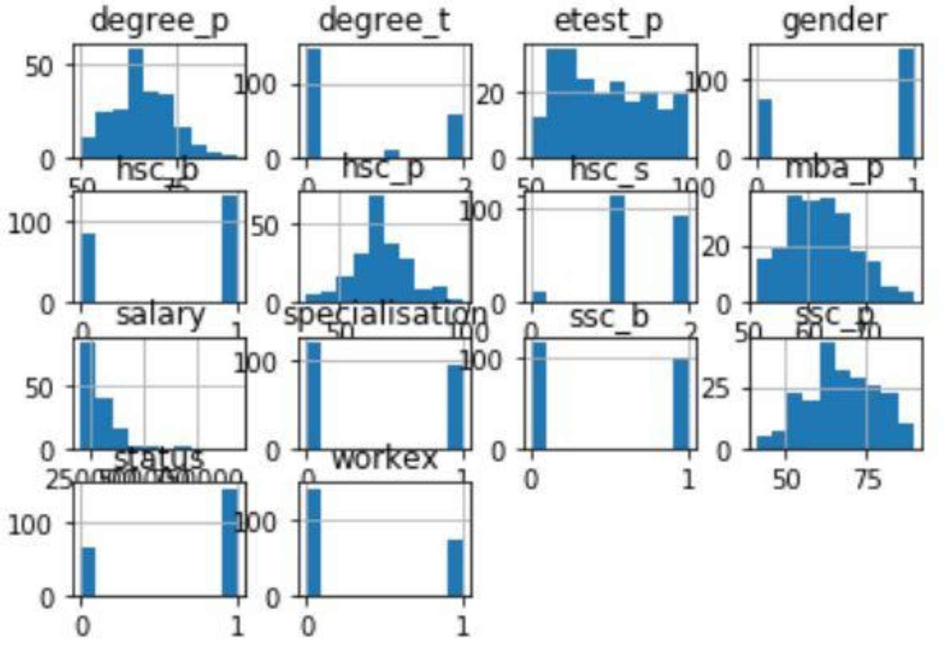
```
import seaborn as sn
from sklearn.metrics import confusion_matrix
conf_matrix =confusion_matrix(prediction,y_test)
confusion_df = pd.DataFrame(conf_matrix, index=['Actual 0','Actual 1'], columns=['Predicted 0','Predicted 1'])
confusion_df
```

	Predicted 0	Predicted 1
Actual 0	19	0
Actual 1	0	46

5.EXPLORATOTRY DATA ANALYSIS (EDA):

5.1] Histograms:

A **histogram** shows the frequency on the vertical axis and the horizontal axis is another dimension. This graph can be plot by using df.hist() function.

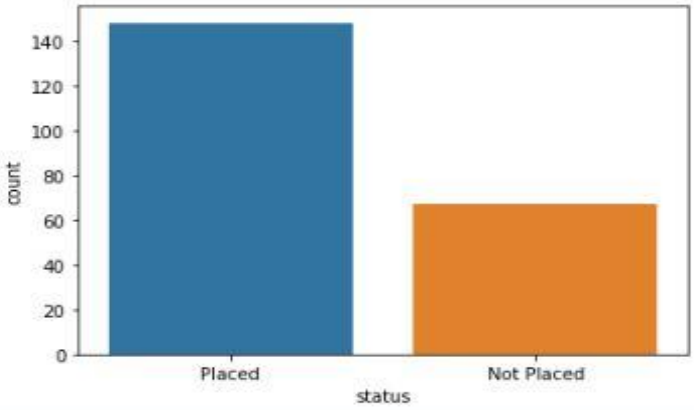


5.2] Bar Plot:

. A **bar chart** or **bar graph** is a **chart** or **graph** that presents categorical data with rectangular **bars** with heights or lengths proportional to the values that they represent. The **bars** can be plotted vertically or horizontally

```
sns.countplot(x='status',data=df)     # Bar plot of Placed and Not Placed students
```

<matplotlib.axes._subplots.AxesSubplot at 0x28a242899c8>

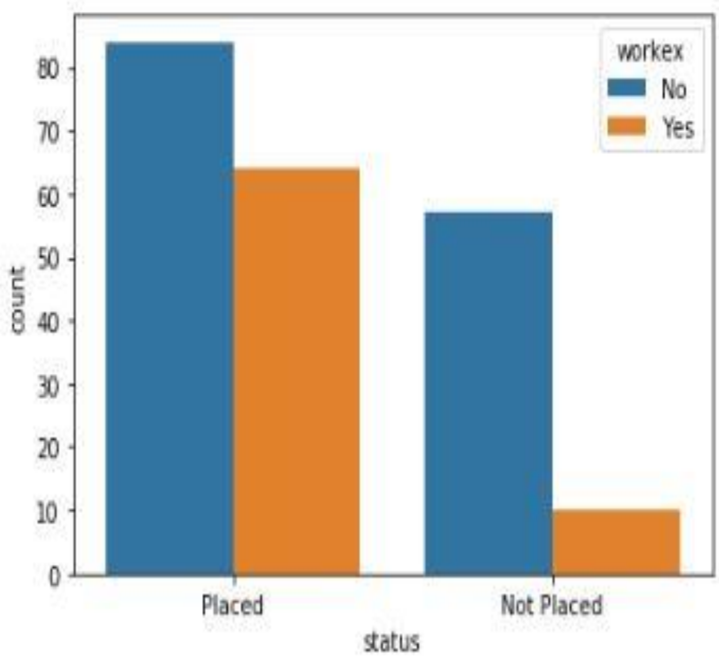


5.3] Grouped Bar Plot:

A **grouped barplot** is used when you have several groups, and subgroups into these groups.

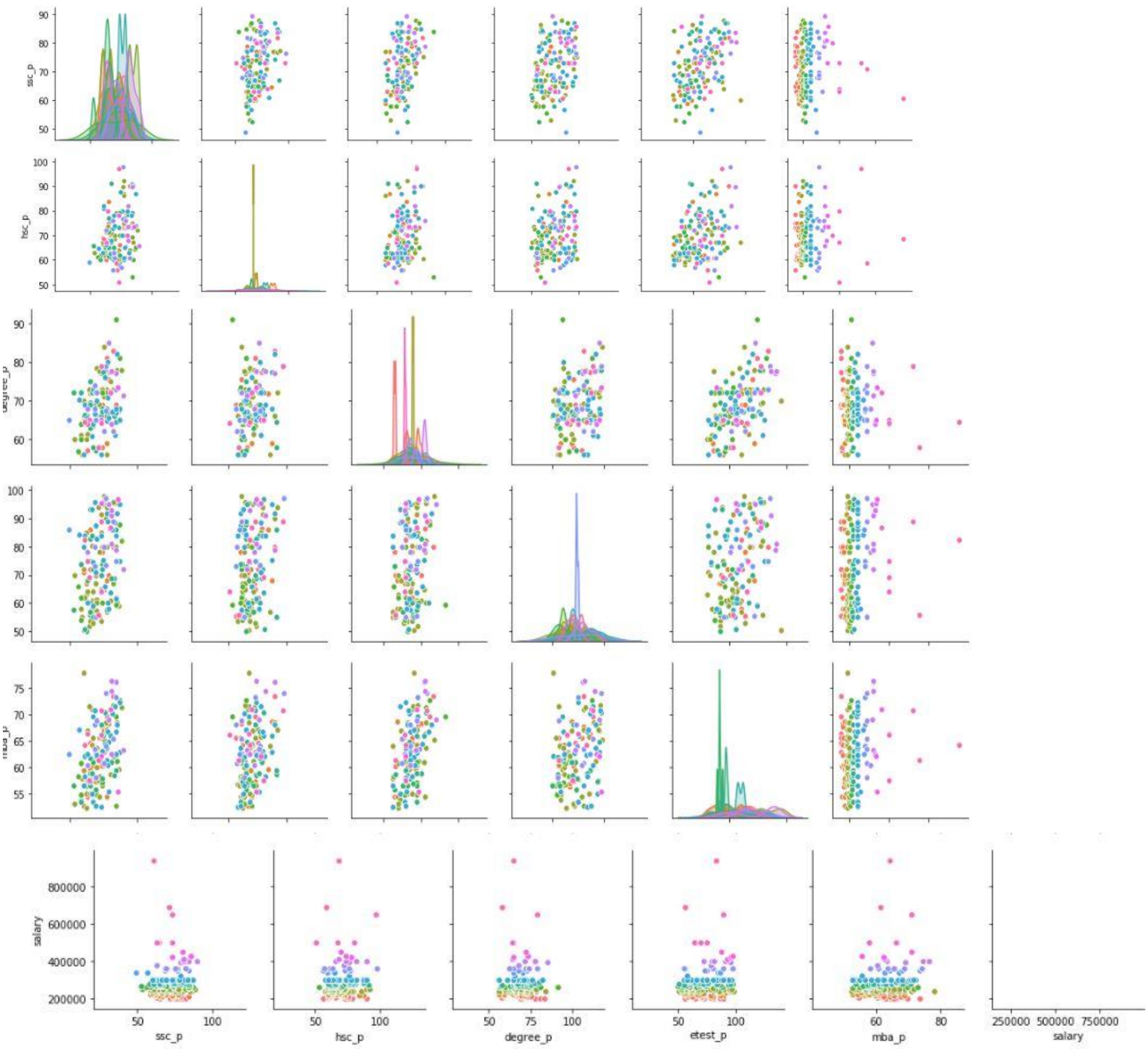
```
sns.countplot(x='status',data=df,hue='workex')     #grouped bar plot of workex and status
```

<matplotlib.axes._subplots.AxesSubplot at 0x28a2413ddc8>



5.4] Pair Plot:

A **pairs plot** allows us to see both distribution of single variables and relationships between two variables . **Pair plots** are a great method to identify trends for follow-up analysis and, fortunately, are easily implemented in **Python**!

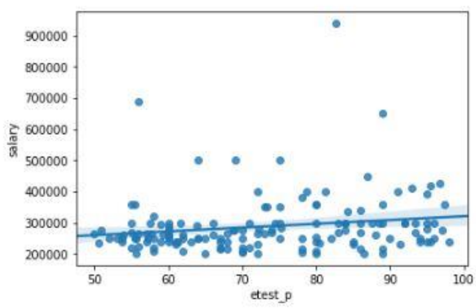


5.5] Regression plot:

The **regression plots** in seaborn are primarily intended to add a visual guide that helps to emphasize patterns in a dataset during exploratory data analyses. **Regression plots** as the name suggests creates a **regression** line between 2 parameters and helps to visualize their linear relationships.

```
sns.regplot(x=df['etest_p'], y=df['salary'])     #regression plot of etest and salary
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x28a22480fc8>
```



etest vs salary 1

6. Training and Test Data

Splitting the Dataset into Training set and Test Set Now the next step is to split our dataset into two. Training set and a Test set. We will train our machine learning models on our training set, i.e. our machine learning models will try to understand any correlations in our training set and then we will test the models on our test set to examine how accurately it will predict. A general rule of the thumb is to assign 80% of the dataset to training set and therefore the remaining 20% to test set. This can be done by using *train_test_split()* function.

```
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)
```

6.1 Algorithms:

6.1.1] Random Forest Classifier:

This classifier takes the concept of decision trees to the next level. It creates a forest of trees where each tree is formed by a random selection of features from the total features

```
from sklearn.ensemble import RandomForestClassifier
model=RandomForestClassifier(n_estimators = 40, criterion = 'entropy', random_state =9)
result=model.fit(X_train,y_train)
```

```
prediction=result.predict(X_test)
prediction
```

```
array([0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0,
       1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1,
       1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1],
      dtype=int64)
```

```
from sklearn import metrics
print('accuracy:',metrics.accuracy_score(y_test,prediction))
```

```
accuracy: 1.0
```

6.1.2] K Nearest Neighbour

This classifier looks for the classes of K nearest neighbors of a given data point and based on the majority class, it assigns a class to this data point. However, the number of neighbors can be varied.

```
from sklearn.neighbors import KNeighborsClassifier
model=KNeighborsClassifier(n_neighbors=3)
result=model.fit(X_train,y_train)
```

```
prediction=result.predict(X_test)
prediction
```

```
array([0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1,
       1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1],
      dtype=int64)
```

```
from sklearn import metrics
print('accuracy:',metrics.accuracy_score(y_test,prediction))
```

```
accuracy: 0.9384615384615385
```

6.1.3] Logistic Regression:

```
from sklearn.linear_model import LogisticRegression
model=LogisticRegression()
result=model.fit(X_train,y_train)
```

```
C:\Users\GK\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:432:
'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
```

```
prediction=result.predict(X_test)
prediction
```

```
array([0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0,
       1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1,
       1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1],
      dtype=int64)
```

```
from sklearn import metrics
print('accuracy:',metrics.accuracy_score(y_test,prediction))
```

```
accuracy: 1.0
```

Here are the Results:

Visualization Method	**Accuracy**
Random Forest Classification	100%
Random Classification	100%
K Nearest Neighbour	93.84%
Logistic Regression	100%
SVM	100%

7.Prediction:

Here I tried one prediction by giving some inputs to te machine and output I got is Not Placed.

```
pred_new = result.predict([[1,0,98,1,98,1,1,78,0,0,99,94.5,1]])
pred_new
```

```
array([0], dtype=int64)
```

8.CONCLUSION:

The campus placement activity is incredibly a lot of vital as institution point of view as well as student point of view. In this regard to improve the student's performance, a work has been analyzed and predicted using the classification algorithms Decision Tree and the Random forest algorithm to validate the approaches. The algorithms are applied on the data set and attributes used to build the model. The accuracy obtained after analysis for K Nearest Neighbour is 93.84% and for the Random Forest is 100%

Here are the Results:

Visualization Method	**Accuracy**
Random Forest Classification	100%
Random Classification	100%
K Nearest Neighbour	93.84%
Logistic Regression	100%
SVM	100%

Hence, from the above said analysis and prediction it's better if the Random Forest algorithm is used to predict the placement result.