PLACEMENT DATA ANALYSIS IN CAMPUS: 1.Exploratory Data Analysis 2.

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

```
#loading data
data=pd.read_csv("/content/pbl_data.csv")
```

Exploratory Data Analysis

Exploring data by each feature

DATA PREPROCESSING

```
data.dtypes
```

```
NAME OF STUDENT (First Name Only)            object
NAME OF STUDENT (Middle Name Only)           object
NAME OF STUDENT (Last Name/Surname Only)     object
Internship Selects                           object
Company Placed Final Offers                  object
Company 1                                    object
Salary (LPA) Company 1                      float64
Salary (LPA) Company 2                      float64
Aggregate (CGPA)                            float64
No. Of live backlogs (e.g - 2)                int64
No. Of Dead backlogs (e. g. - 4)              int64
Unnamed: 11                                 float64
Gender                                       object
status                                       object
dtype: object
```

```
data.head()
```

	NAME OF STUDENT (First Name Only)	NAME OF STUDENT (Middle Name Only)	NAME OF STUDENT (Last Name/Surname Only)	Internship Selects	Company Placed Final Offers	Company 1	Salary (LPA) Company 1	Salary (LPA) Company 2	Aggregate (CGPA)	No. Of live backlogs (e.g - 2)	No. Of Dead backlogs (e. g. - 4)	Unnamed: 11	Ger
0	Nilesh	Sunil	Kudale	Persistent	NaN	NaN	0.0	0.0	9.80	0	0	NaN	r
1	Janhavi	Tarachand	Bhondge	NaN	NaN	NaN	0.0	0.0	8.84	0	0	NaN	fer
2	Vishwatej	Vitthal	Katkar	NaN	NaN	NaN	0.0	0.0	8.58	0	0	NaN	fer
3	Virajas	Vikas	Joshi	Kinetic Communications	NaN	NaN	0.0	0.0	8.90	1	1	NaN	r
4	Rajat	Prakash	Madyapgol	NaN	NaN	NaN	0.0	0.0	7.54	0	0	NaN	r

Next steps: View recommended plots

1.GENDER

Does gender affect placements?

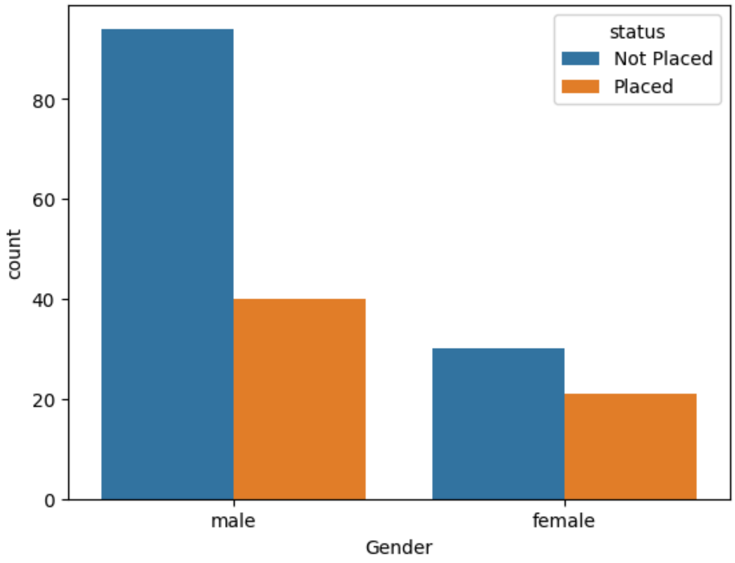
```
data.Gender.value_counts()
```

```
Gender
male      134
female     51
Name: count, dtype: int64
```

description:

the number of males getting placed is more than twice than female

```
#Gender v/s Status
sns.countplot(x="Gender", hue="status", data=data)
plt.show()
```



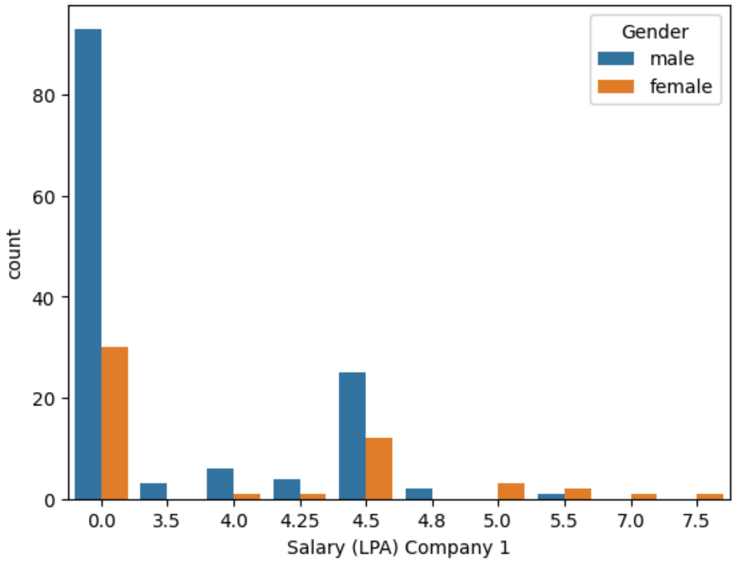
INSIGHTS:

1.We have samples of 134 males out of which 40 are placed and 94 are not placed

2.Out of 51 female students 20 are placed and 31 are unplaced

2.CGPA

```
#Gender V/S Salary
sns.countplot(x="Salary (LPA) Company 1", hue="Gender", data=data)
plt.show()
```

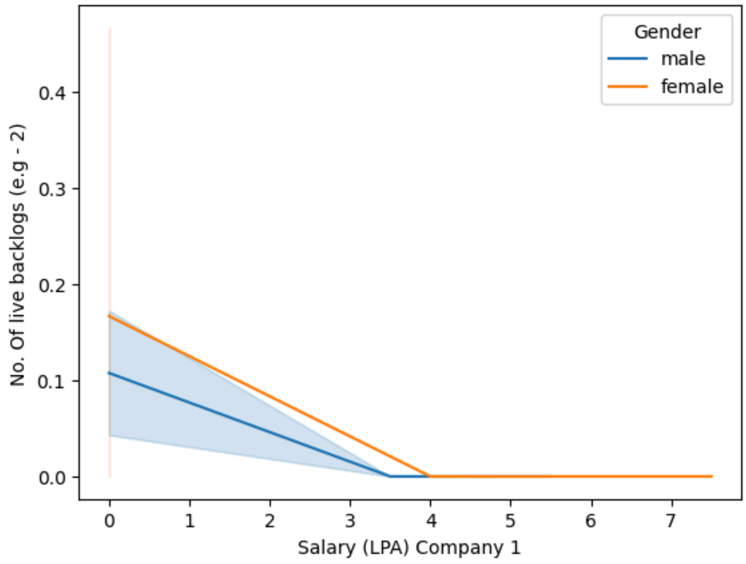


Insights

1.Overall the number of male candidates placed are more than that of the female candidates

2.The female candidates are observed to have the highest packages than that of the males .

```
#Live backlogs V/S Salary
sns.lineplot(x='Salary (LPA) Company 1', y='No. Of live backlogs (e.g - 2)',hue='Gender',data=data)
plt.show()
```



Insights

1.The male and female candidates with 0 live lacklogs has the highest package

2.As the number of backlogs increases the Package decreases

```
#FINAL COMBINED OFFERS  V/S PLACED COUNT

# Grouping by a column containing string values
grouped_data = data.groupby('Company Placed Final Offers')

# You can then perform operations on each group, for example:
# Calculating the count of rows in each group
grouped_count = grouped_data.size()

# Displaying the count of rows in each group
print(grouped_count)
```

```
Company Placed Final Offers
Accenture                           12
Acuiti Labs                          1
Dassault Systemes, Accenture         1
HPE                                  3
ITC Infotech                         5
KPIT                                19
KPIT, Accenture                      6
Keyence                              2
Keyence, Accenture                   1
Newgen                               3
Rudder Analytics                     2
Tata Elxsi                           3
Tech Mahindra                        1
Tech Mahindra, Accenture             2
Tech Mahindra, Dassault Systemes     1
dtype: int64
```

```
#individual company v/s count of placed

# Grouping by a column containing string values
grouped_data = data.groupby('Company 1')

# You can then perform operations on each group, for example:
# Calculating the count of rows in each group
grouped_count = grouped_data.size()

# Displaying the count of rows in each group
print(grouped_count)
```

```
Company 1
Accenture           12
Acuiti Labs          1
Dassault Systemes    1
HPE                  3
ITC Infotech         5
KPIT                25
Keyence              3
Newgen               3
Rudder Analytics     2
```

```
     Tata Elxsi            3
     Tech Mahindra         4
     dtype: int64
```

Insights:

1.The company named "KPIT" has been a mass recruiter in 2023-24 with a total recruitment of 25 candidates

2.The second mass recruiter is "Accenture" with a mass recruitment of 12 candidates

```
#

# Grouping by a column containing string values and calculating the mean of another numerical column
grouped_data1 = data.groupby('Company 1')['Salary (LPA) Company 1'].mean()

# Grouping by a column containing string values
grouped_data = data.groupby('Company 1')

# Calculating the count of rows in each group
grouped_count = grouped_data.size()

# Displaying the count of rows in each group
print(grouped_count)

# Displaying the mean values for each group
print(grouped_data1)
```

```
     Company 1
     Accenture            12
     Acuiti Labs           1
     Dassault Systemes     1
     HPE                   3
     ITC Infotech          5
     KPIT                 25
     Keyence               3
     Newgen                3
     Rudder Analytics      2
     Tata Elxsi            3
     Tech Mahindra         4
     dtype: int64
     Company 1
     Accenture            4.50
     Acuiti Labs          7.00
     Dassault Systemes    7.50
     HPE                  5.00
     ITC Infotech         4.25
     KPIT                 4.50
     Keyence              5.50
     Newgen               4.00
     Rudder Analytics     4.80
     Tata Elxsi           3.50
     Tech Mahindra        4.00
     Name: Salary (LPA) Company 1, dtype: float64
```

Insights:

1.The company with less count of recruitment is observed to be the highest paying companies

2.KPIT has proved to be a mass recruiter with the package of 4.5 lakh

MODEL ANALYSIS USING VARIOUS ALGORITHMS:

```
# Library imports
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report
```

```
#Lets make a copy of data, before we proceeed with specific problems
data_clf = data.copy()
data_reg = data.copy()
X = data_clf[['Aggregate (CGPA)','No. Of live backlogs (e.g - 2)']]
y = data_clf['status']
```

```
# Replace NaN values by zero in a particular column (e.g., 'column_name')
data['Company 1'] = data['Company 1'].fillna(0)
data['No. Of live backlogs (e.g - 2)'] = data['No. Of live backlogs (e.g - 2)'].fillna(0)
```

```
#Train Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,random_state=42)
```

```
dtree = DecisionTreeClassifier(criterion='entropy')
dtree.fit(X_train, y_train)
y_pred = dtree.predict(X_test)
```

```
accuracy_score(y_test, y_pred)
```

```
    0.6216216216216216
```

```
print(classification_report(y_test, y_pred))
```

```
                   precision    recall  f1-score   support

      Not Placed       0.62      0.80      0.70        20
          Placed       0.64      0.41      0.50        17

        accuracy                           0.62        37
       macro avg       0.63      0.61      0.60        37
    weighted avg       0.63      0.62      0.61        37
```

```
#Using Random Forest Algorithm
random_forest = RandomForestClassifier(n_estimators=100)
random_forest.fit(X_train, y_train)
y_pred = random_forest.predict(X_test)
```

```
accuracy_score(y_test, y_pred)
```

```
    0.6216216216216216
```

```
print(classification_report(y_test, y_pred))
```

```
                   precision    recall  f1-score   support

      Not Placed       0.62      0.80      0.70        20
          Placed       0.64      0.41      0.50        17

        accuracy                           0.62        37
       macro avg       0.63      0.61      0.60        37
    weighted avg       0.63      0.62      0.61        37
```

```
#Analysis using Logistic regression

# Seperating Features and Target
X = data_clf[['Aggregate (CGPA)','No. Of live backlogs (e.g - 2)']]
y = data_clf['status']
```

```
#One-Hot Encoding
X = pd.get_dummies(X)
colmunn_names = X.columns.to_list()
```

```
#*Scaling Everything between 0 and 1 (This wont affect one-hot encoded values)**

from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
X_scaled = scaler.fit_transform(X)
#Train Test Split
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.3)
```

```
from sklearn.linear_model import LogisticRegression
logistic_reg = LogisticRegression()
logistic_reg.fit(X_train, y_train)
y_pred = logistic_reg.predict(X_test)
```

```
accuracy_score(y_test, y_pred)
```

```
    0.6964285714285714
```

```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

  Not Placed       0.70      1.00      0.82        39
      Placed       0.00      0.00      0.00        17

    accuracy                           0.70        56
   macro avg       0.35      0.50      0.41        56
weighted avg       0.49      0.70      0.57        56

/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are
  _warn_prf(average, modifier, msg_start, len(result))
```

```
#K nearest neighbour
# Importing necessary libraries
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Assuming 'X' is your feature matrix and 'y' is your target vector

# Splitting the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Instantiating the KNeighborsClassifier object
knn = KNeighborsClassifier(n_neighbors=5)  # You can specify the number of neighbors (k) with the n_neighbors parameter

# Fitting the model with the training data
knn.fit(X_train, y_train)

# Making predictions with the testing data
y_pred = knn.predict(X_test)

# Evaluating the model
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.5945945945945946
```

```
#Analysis using Naive Bayes
# Importing necessary libraries
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Assuming 'X' is your feature matrix and 'y' is your target vector

# Splitting the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Instantiating the Gaussian Naive Bayes object
naive_bayes = GaussianNB()

# Fitting the model with the training data
naive_bayes.fit(X_train, y_train)

# Making predictions with the testing data
y_pred = naive_bayes.predict(X_test)

# Evaluating the model
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.5135135135135135
```

```
data = {
    'Algorithm': ['Logistic Regression', 'Decision Tree', 'KNN','Naive Bayes'],
    'Accuracy': [0.76, 0.75, 0.59,0.52]
}

df = pd.DataFrame(data)
```

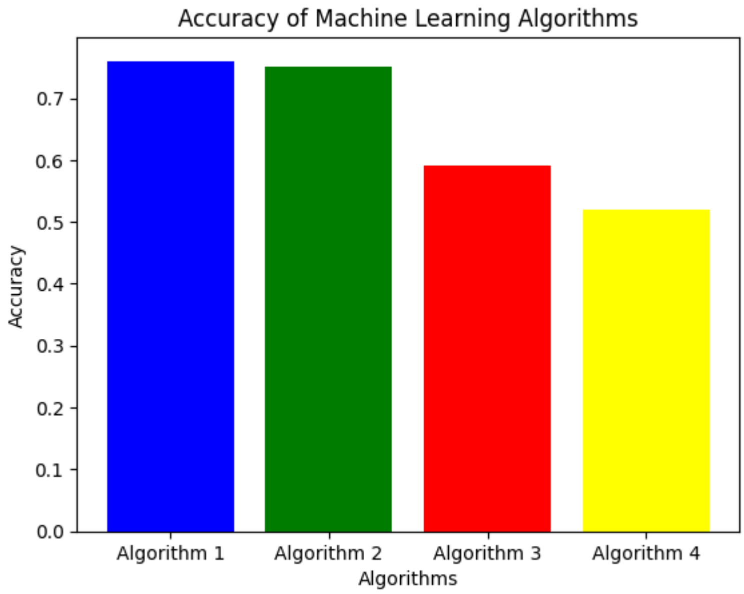
```
import matplotlib.pyplot as plt

# Define the machine learning algorithms
algorithms = ['Algorithm 1', 'Algorithm 2', 'Algorithm 3','Algorithm 4']
algorithms = ['Algorithm 1', 'Algorithm 2', 'Algorithm 3', 'Algorithm 4']
accuracy = [0.76, 0.75, 0.59,0.52]  # Replace these values with your actual accuracy scores

# Create the bar plot
plt.bar(algorithms, accuracy, color=['blue', 'green', 'red','yellow'])

# Add title and labels
plt.title('Accuracy of Machine Learning Algorithms')
plt.xlabel('Algorithms')
plt.ylabel('Accuracy')

# Show the plot
plt.show()
```


Accuracy of Machine Learning Algorithms
Accuracy
Algorithm 1
Algorithm 2
Algorithm 3
Algorithm 4
Algorithms