# PRACTICAL 6

AIM: Apply Logistic Regression, K-nearest- neighbors, Naive Bayes and Support Vector Machine (SVM) on given Social_Network_Ads.csv dataset for classification.

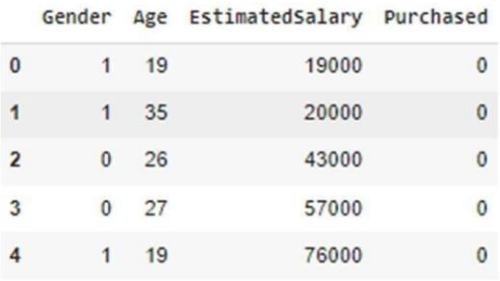
## Read Dataset

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

```
data =  pd.read_csv("/content/drive/MyDrive/ML_Collage/Social_Ne
twork_Ads.csv")
```

## Data analysis

```
data.head()
```

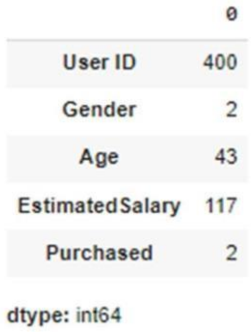


| | Gender | Age | EstimatedSalary | Purchased |
|---|---|---|---|---|
| 0 | 1 | 19 | 19000 | 0 |
| 1 | 1 | 35 | 20000 | 0 |
| 2 | 0 | 26 | 43000 | 0 |
| 3 | 0 | 27 | 57000 | 0 |
| 4 | 1 | 19 | 76000 | 0 |

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'> RangeIndex: 400 entries, 0 to 399
Data columns (total 5 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
0       User ID          400 non-null    int64
1       Gender           400 non-null    object
2       Age            400 non-null    int64
3       EstimatedSalary  400 non-null    int64   4   Purchased        400 nonnull
        int64  dtypes: int64(4), object(1) memory usage: 15.8+ KB
```

```
data.nunique()
```



```
                  0
User ID         400
Gender            2
Age              43
EstimatedSalary 117
Purchased         2

dtype: int64
```

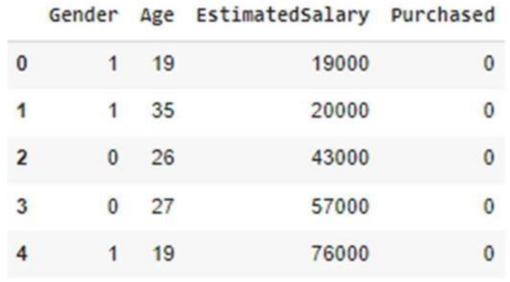
```
data.isnull().any(axis=1).sum()
```

```
0
```

## Pre-processing

```
data.drop(columns="User ID", inplace=True)
```

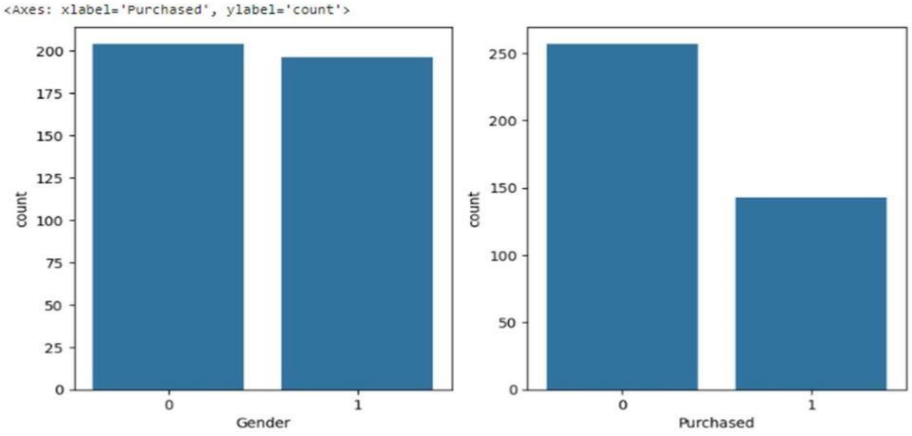
```
from sklearn.preprocessing import LabelEncoder label = LabelEncoder()
data["Gender"] = label.fit_transform(data["Gender"])   data.head()
```



```
   Gender  Age  EstimatedSalary  Purchased
0       1   19            19000          0
1       1   35            20000          0
2       0   26            43000          0
3       0   27            57000          0
4       1   19            76000          0
```

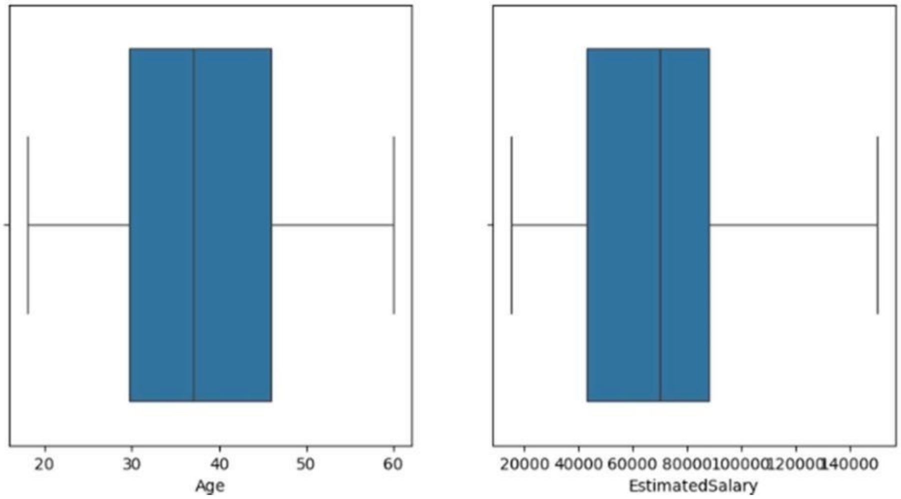
## Data visualization

```
fig, ax = plt.subplots(1, 2, figsize=(10,5)) sns.countplot(data=data, x="Gender",
ax=ax[0]) sns.countplot(data=data, x="Purchased", ax=ax[1])
```

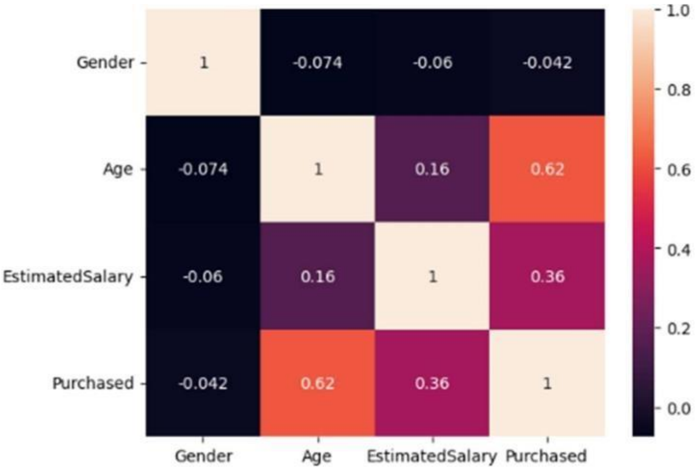


```
<Axes: xlabel='Purchased', ylabel='count'>
```

```
fig, ax  = plt.subplots(1, 2, figsize=(10,5)) sns.boxplot(data=data, x="Age",
ax=ax[0]) sns.boxplot(data=data, x="EstimatedSalary", ax=ax[1])
```



```
sns.heatmap(data.corr(), annot=True)
```



## Data Splitting

```
from sklearn.model_selection import train_test_split x =
data.drop(columns="Purchased") y = data["Purchased"]  x_train,
x_test, y_train, y_test = train_test_split(x, y, test_size=0.3,
random_state=42)
```

## Feature Scaling

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

## Different Models

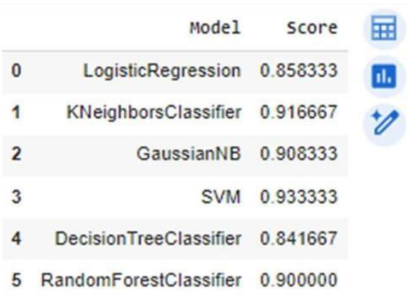
```
from sklearn.linear_model import LogisticRegression from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier from sklearn.neighbors
import KNeighborsClassifier from sklearn.naive_bayes import GaussianNB from
sklearn.tree import DecisionTreeClassifier


dict = {
   "LogisticRegression": LogisticRegression(),
   "KNeighborsClassifier":
KNeighborsClassifier(n_neighbors=5),
   "GaussianNB": GaussianNB(),
   "SVM": SVC(),
   "DecisionTreeClassifier": DecisionTreeClassifier(),
   "RandomForestClassifier": RandomForestClassifier()
}
model_list = [] score_list = []
```

```
from sklearn.metrics import accuracy_score,
confusion_matrix, classification_report for i, j in dict.items():
model = j
  model.fit(x_train, y_train)   pred =
model.predict(x_test)

  model_list.append(i)
  score_list.append(accuracy_score(y_test, pred))
```

```
result = pd.DataFrame({"Model": model_list,"Score":  score_list})
result
```



|   | Model | Score |
|---|---|---|
| 0 | LogisticRegression | 0.858333 |
| 1 | KNeighborsClassifier | 0.916667 |
| 2 | GaussianNB | 0.908333 |
| 3 | SVM | 0.933333 |
| 4 | DecisionTreeClassifier | 0.841667 |
| 5 | RandomForestClassifier | 0.900000 |