Practice KNN - We have a dataset that contains multiple user's information through the social network who are interested in buying SUV Car or not.

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

```
df=pd.read_csv('/content/User_Data.csv')
df.head()
```

	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0

Exploratory Data Analysis

- Handling missing values if any
- Apply label encoding to convert categorical data into numerical
- Handle outliers
- Visualization

```
df.shape
```

```
(400, 5)
```

```
df.duplicated().sum()
```

```
0
```

```
df.isnull().sum()
```

```
User ID            0
Gender             0
Age                0
```

```
EstimatedSalary    0
Purchased          0
dtype: int64
```

```
df.dtypes
```

```
User ID             int64
Gender             object
Age                 int64
EstimatedSalary     int64
Purchased           int64
dtype: object
```

```
df.columns
```

```
Index(['User ID', 'Gender', 'Age', 'EstimatedSalary', 'Purchased'], dtype='object')
```

```
df.corr()
```

	User ID	Age	EstimatedSalary	Purchased
User ID	1.000000	-0.000721	0.071097	0.007120
Age	-0.000721	1.000000	0.155238	0.622454
EstimatedSalary	0.071097	0.155238	1.000000	0.362083
Purchased	0.007120	0.622454	0.362083	1.000000

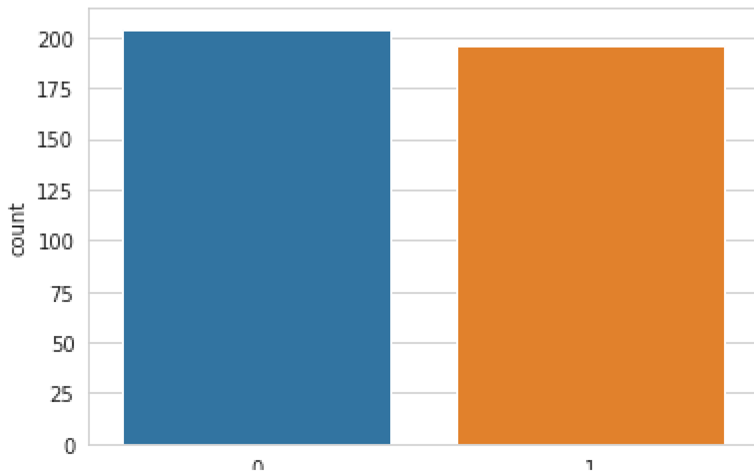
Label Encoding

```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
df.Gender=le.fit_transform(df.Gender)
df.Gender.head()
```

```
0    1
1    1
2    0
3    0
4    1
Name: Gender, dtype: int64
```

Data Visualization

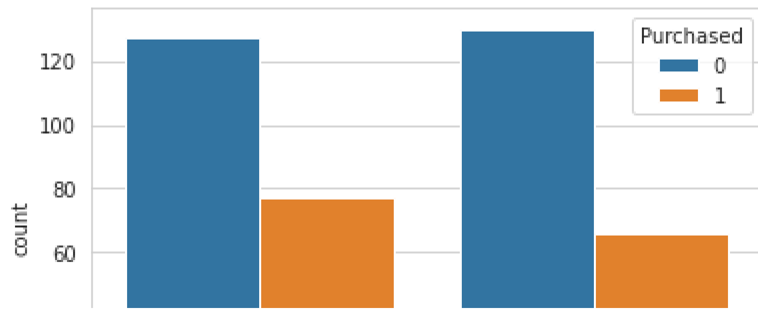
```
sns.countplot(x = "Gender", data = df);
```



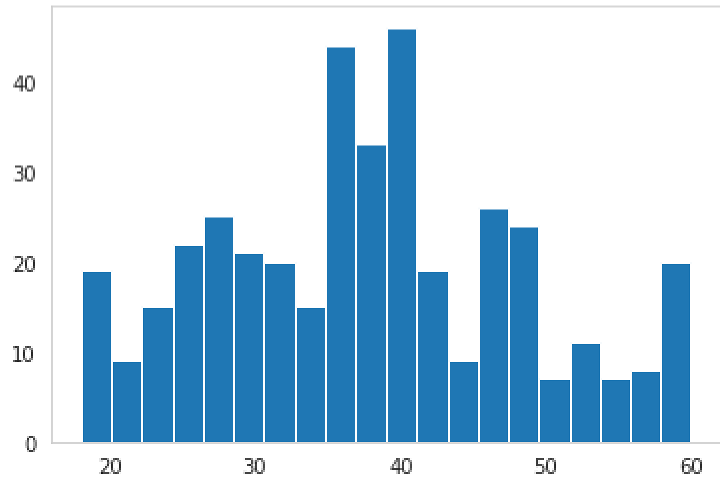

```
x=df.drop(['Purchased','User ID',],axis='columns')
print(x)
y=df['Purchased']
print(y)
```

```
     Gender  Age  EstimatedSalary
0         1   19            19000
1         1   35            20000
2         0   26            43000
3         0   27            57000
4         1   19            76000
..      ...  ...              ...
395       0   46            41000
396       1   51            23000
397       0   50            20000
398       1   36            33000
399       0   49            36000

[400 rows x 3 columns]
0      0
1      0
2      0
3      0
4      0
      ..
395    1
396    1
397    1
398    0
399    1
Name: Purchased, Length: 400, dtype: int64
```

```
sns.countplot(x = "Gender", hue = "Purchased", data = df);
```

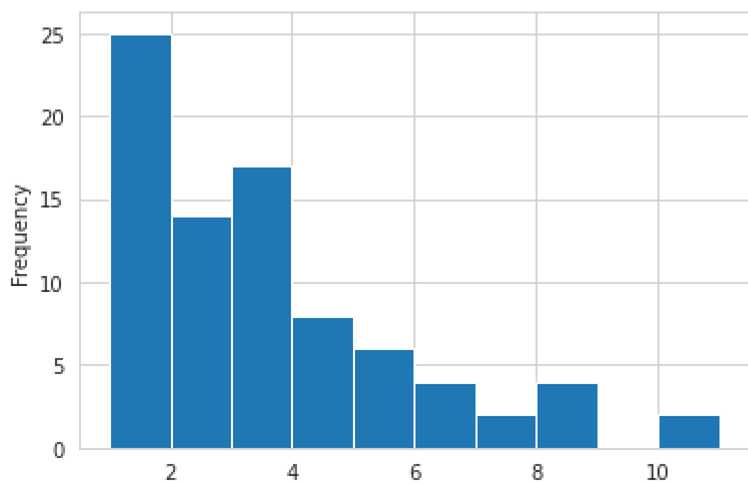



```
plt.grid()
plt.hist(x = df["Age"], bins = 20);
```



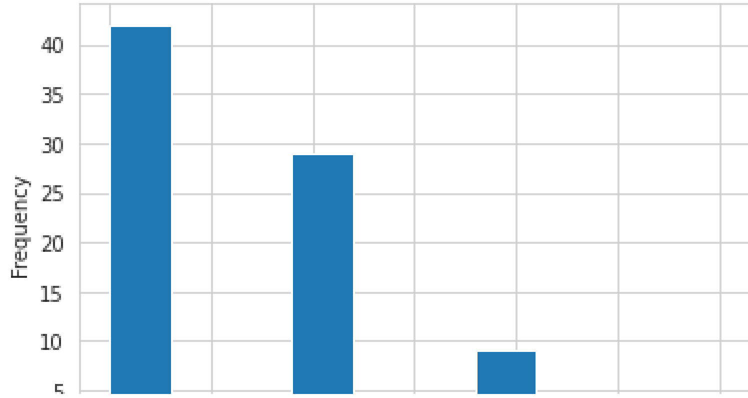

```
purch_0 = df[df["Purchased"] == 0].groupby(["EstimatedSalary"]).count()
purch_0["Purchased"].plot(kind = "hist")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f3009b94e50>
```

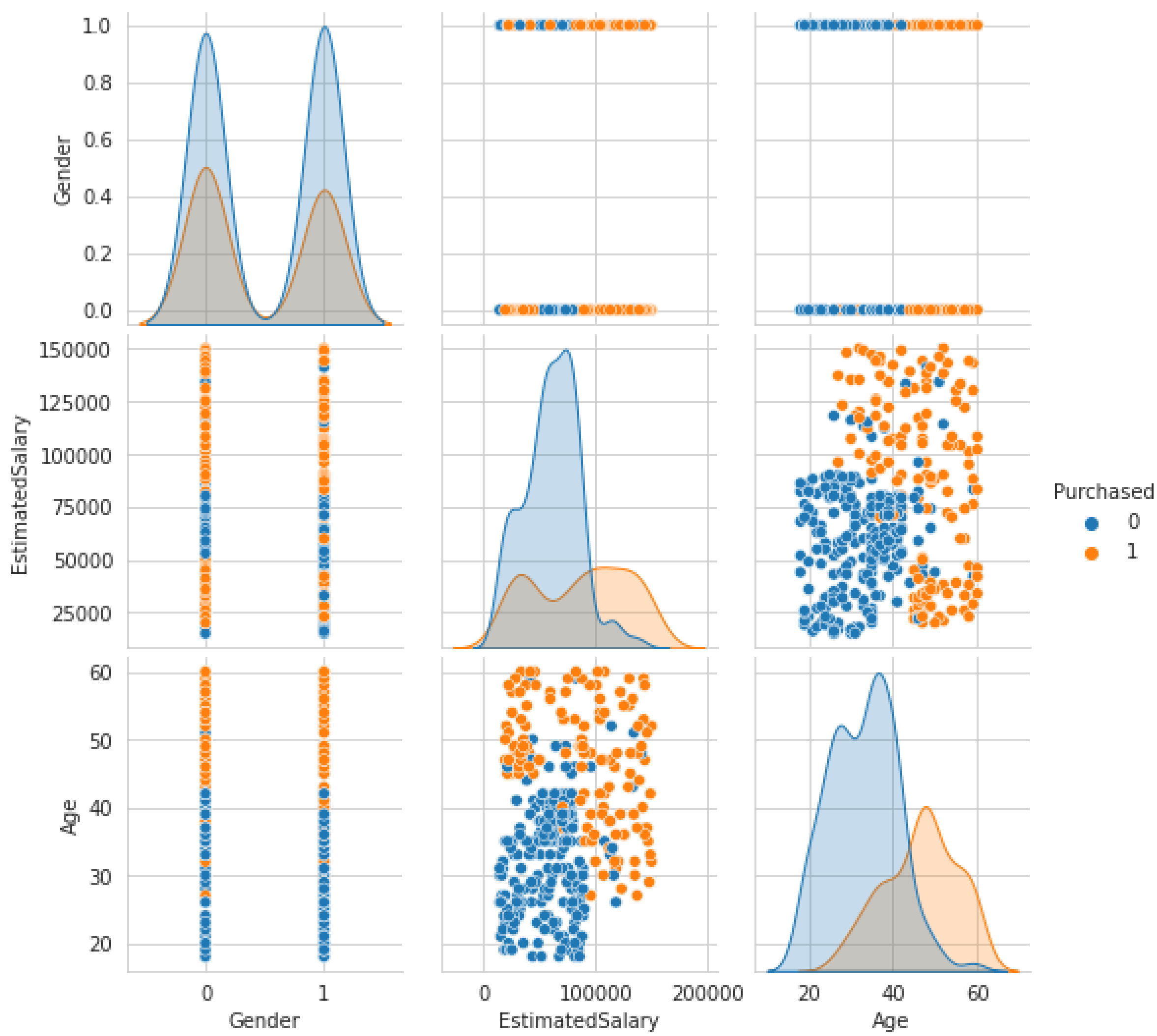



```
purch_1 = df[df["Purchased"] == 1].groupby(["EstimatedSalary"]).count()
purch_1["Purchased"].plot(kind = "hist")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f3009b187d0>
```




```
sns.pairplot(data = df, hue = "Purchased", vars = ["Gender", "EstimatedSalary", "Age"]);
```



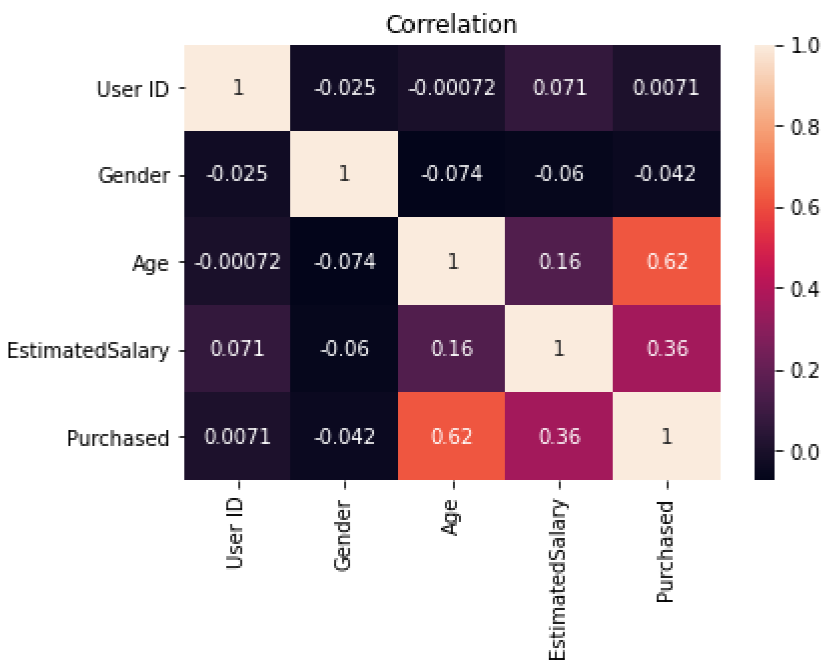

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
x.drop('Gender',axis=1,inplace=True)
x=scaler.fit_transform(x)
print(x)
```

```
[[-1.78179743 -1.49004624]
 [-0.25358736 -1.46068138]
 [-1.11320552 -0.78528968]
 [-1.01769239 -0.37418169]
 [-1.78179743  0.18375059]
 [-1.01769239 -0.34481683]
 [-1.01769239  0.41866944]
 [-0.54012675  2.35674998]
 [-1.20871865 -1.07893824]
 [-0.25358736 -0.13926283]
 [-1.11320552  0.30121002]
 [-1.11320552 -0.52100597]
 [-1.6862843   0.47739916]
 [-0.54012675 -1.51941109]
 [-1.87731056  0.35993973]
 [-0.82666613  0.30121002]
 [ 0.89257019 -1.3138571 ]
 [ 0.70154394 -1.28449224]
 [ 0.79705706 -1.22576253]
 [ 0.98808332 -1.19639767]
 [ 0.70154394 -1.40195167]
 [ 0.89257019 -0.60910054]
 [ 0.98808332 -0.84401939]
 [ 0.70154394 -1.40195167]
 [ 0.79705706 -1.37258681]
 [ 0.89257019 -1.46068138]
 [ 1.08359645 -1.22576253]
 [ 0.89257019 -1.16703281]
 [-0.82666613 -0.78528968]
 [-0.63563988 -1.51941109]
 [-0.63563988  0.12502088]
 [-1.01769239  1.97500684]
 [-1.59077117 -1.5781408 ]
 [-0.92217926 -0.75592482]
 [-1.01769239  0.59485858]
 [-0.25358736 -1.25512738]
 [-0.44461362 -1.22576253]
 [-0.73115301 -0.60910054]
 [-1.11320552  0.06629116]
 [-1.01769239 -1.13766796]
 [-1.01769239 -1.54877595]
 [-0.44461362 -0.55037082]
 [-0.25358736  1.123426  ]
 [-0.73115301 -1.60750566]
 [-0.92217926  0.41866944]
 [-1.39974491 -1.46068138]
 [-1.20871865  0.27184516]
 [-1.01769239 -0.46227625]
 [-0.73115301  1.91627713]
 [-0.63563988  0.56549373]
 [-1.30423178 -1.1083031 ]
 [-1.87731056 -0.75592482]
 [-0.82666613  0.38930459]
 [-0.25358736 -1.37258681]
 [-1.01769239 -0.34481683]
 [-1.30423178 -0.4329114 ]
```

```
[-1.39974491 -0.63846539]
[-0.92217926  0.27184516]
[-1.49525804 -1.51941109]
```

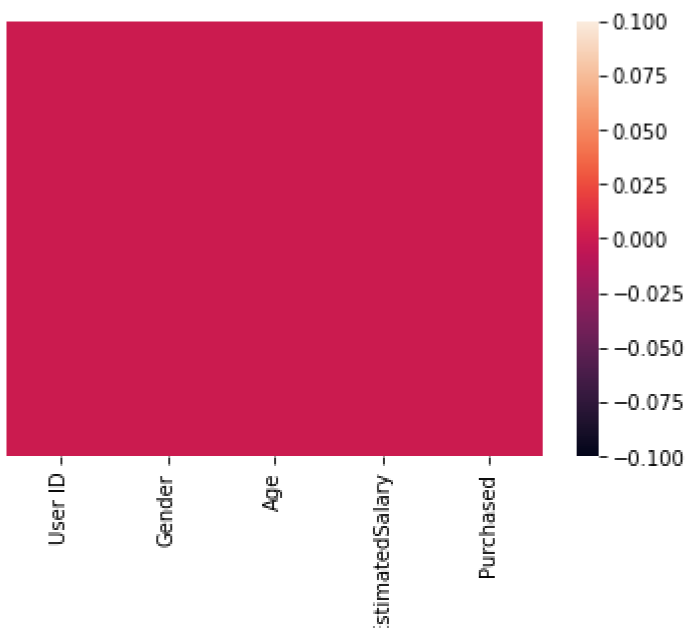
Visualization

```
sns.heatmap(df.corr(),annot=True)
plt.title('Correlation')
plt.show()
```



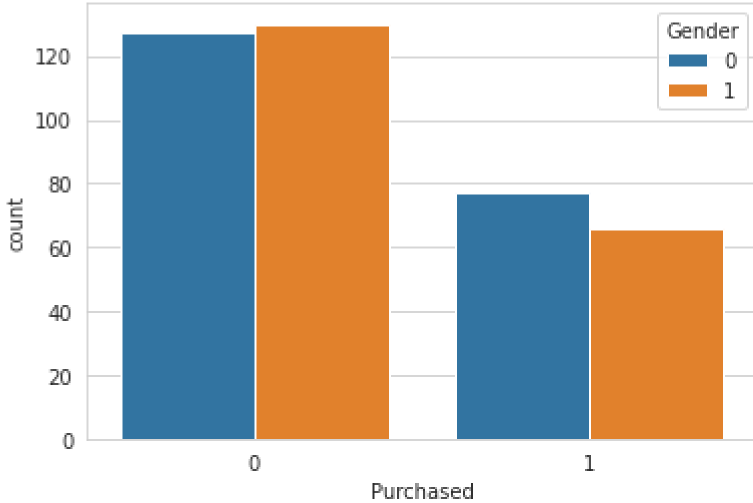

```
sns.heatmap(df.isnull(),yticklabels=False)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f3011e331d0>
```



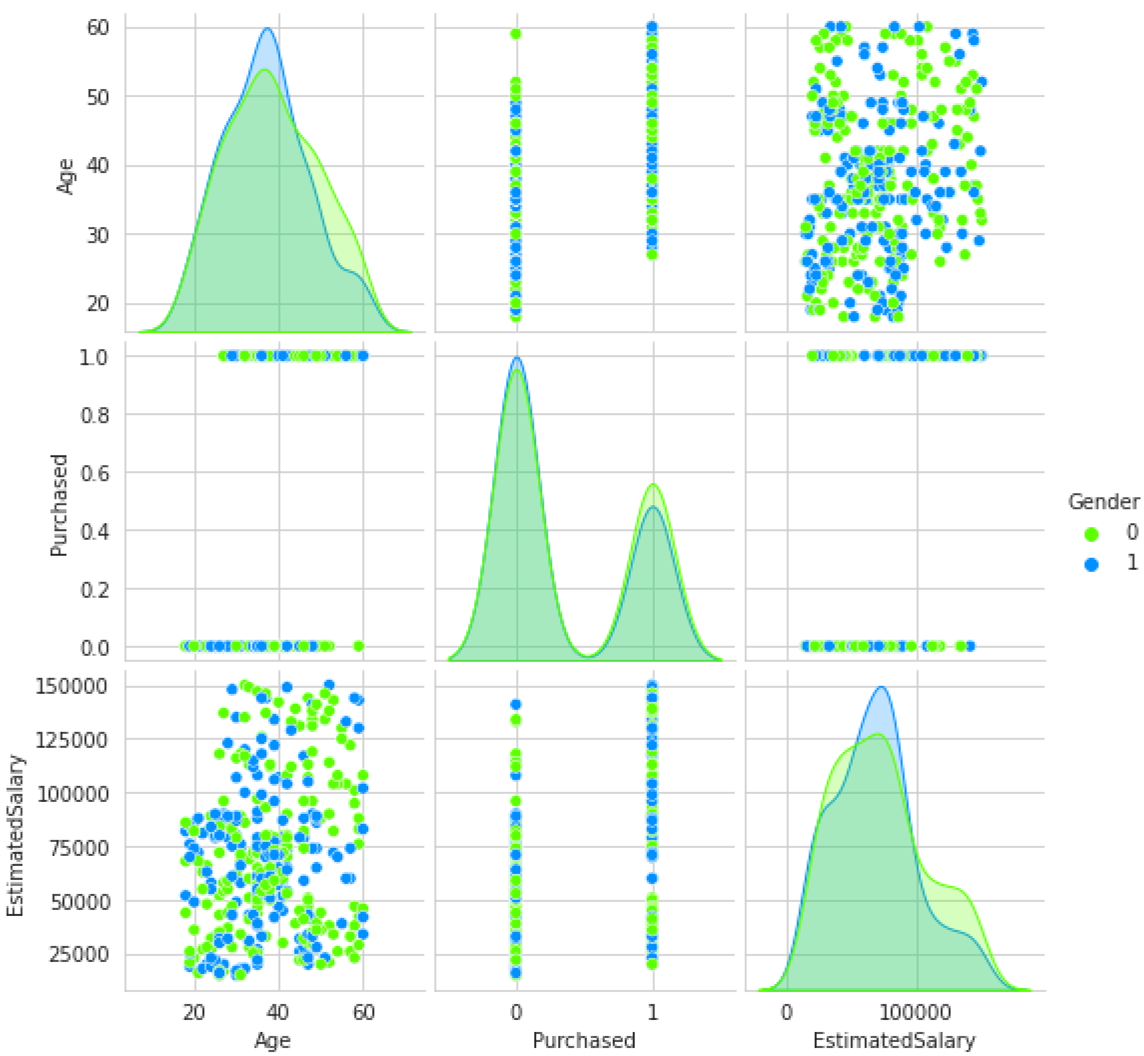
```
sns.set_style('whitegrid')
sns.countplot(x='Purchased',hue='Gender',data=df)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f3011699610>
```




```
sns.pairplot(df,hue='Gender',vars=['Age','Purchased','EstimatedSalary'],palette='gist_rainbow
```

```
<seaborn.axisgrid.PairGrid at 0x7f301060cf50>
```



Detecting Outliers

```
max_threshold=df['Age'].quantile(0.95)
print(max_threshold)
min_threshold=df['Age'].quantile(0.05)
print(min_threshold)
```

```
57.049999999999955
21.0
```

```
df[df['Age']>max_threshold]
```

	User ID	Gender	Age	EstimatedSalary	Purchased
64	15605000	0	59	83000	0
204	15660866	0	58	101000	1
212	15707596	0	59	42000	0
215	15779529	0	60	108000	1
219	15732987	1	59	143000	1
223	15593715	1	60	102000	1
258	15569641	0	58	95000	1
271	15688172	0	59	76000	1
272	15791373	1	60	42000	1
280	15609669	0	59	88000	1
300	15736397	0	58	38000	1
336	15664907	1	58	144000	1
355	15606472	1	60	34000	1
365	15807525	0	59	29000	1
366	15574372	0	58	47000	1
370	15611430	0	60	46000	1
371	15774744	1	60	83000	1
373	15708791	1	59	130000	1
379	15749381	0	58	23000	1
393	15635893	1	60	42000	1

```
df[df['Age']<min_threshold]
```

	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	1	19	19000	0
4	15804002	1	19	76000	0
12	15746139	1	20	86000	0
14	15628972	1	18	82000	0
51	15764195	0	18	44000	0
72	15595228	0	20	23000	0
76	15746737	1	18	52000	0
82	15709476	1	20	49000	0
104	15672091	0	19	21000	0
136	15668504	0	20	82000	0
139	15741094	1	19	25000	0
140	15807909	1	19	85000	0
141	15666141	0	18	68000	0
149	15767871	1	20	74000	0
165	15578738	0	18	86000	0
186	15724402	0	20	82000	0
191	15662067	0	19	26000	0
193	15662901	1	19	70000	0
197	15680243	0	20	36000	0

Removing Outliers

```
df[(df['Age']<max_threshold)&(df['Age']>min_threshold)]
```

	User ID	Gender	Age	EstimatedSalary	Purchased
1	15810944	1	35	20000	0
2	15668575	0	26	43000	0
3	15603246	0	27	57000	0
5	15728773	1	27	58000	0
6	15598044	0	27	84000	0

Splitting Data into Train and Test

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=10)
```

KNN Classifier Model

```
from sklearn.neighbors import KNeighborsClassifier
KNN=KNeighborsClassifier(n_neighbors=3)
KNN.fit(x_train,y_train)
```

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=3, p=2,
                     weights='uniform')
```

Prediction

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

```
array([0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0,
       0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0,
       0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0,
       0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1])
```

Confusion Matrix

```
from sklearn.metrics import confusion_matrix, accuracy_score
```

```
matrix = confusion_matrix(y_test, y_pred)
matrix
```

```
array([[46,  6],
       [ 1, 27]])
```

```
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy of model: {}%".format(accuracy*100))
```

```
Accuracy of model: 91.25%
```

Classification Report

```
from sklearn.metrics import classification_report
```

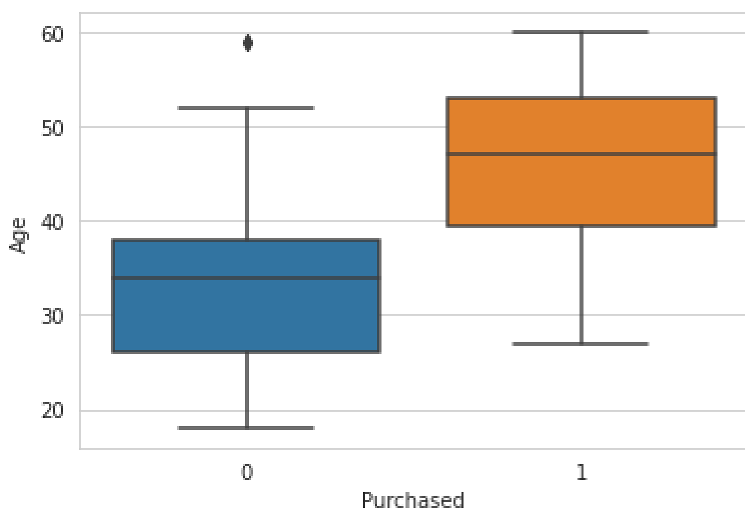
```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.98      0.88      0.93        52
           1       0.82      0.96      0.89        28

    accuracy                           0.91        80
   macro avg       0.90      0.92      0.91        80
weighted avg       0.92      0.91      0.91        80
```

Box Plot

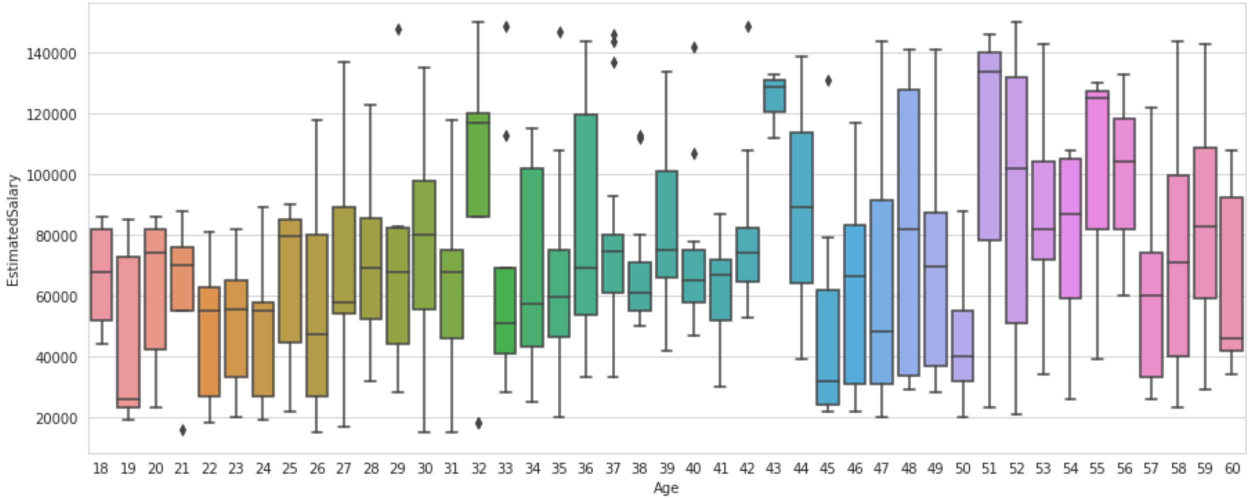
```
sns.boxplot(x='Purchased',y='Age',data=df)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f300972e1d0>
```




```
plt.figure(figsize=(15,6))
sns.boxplot(x='Age',y='EstimatedSalary',data=df)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f3009704f90>
```



Visualization -Accuracy Score

```
plt.figure(figsize=(5,5))
sns.heatmap(matrix, annot=True, fmt=".2f", linewidths=.5, square = True, cmap = 'Blues_r')
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
A=f'Accuracy Score :{accuracy:.2f}'
plt.title(A)
plt.show()
```