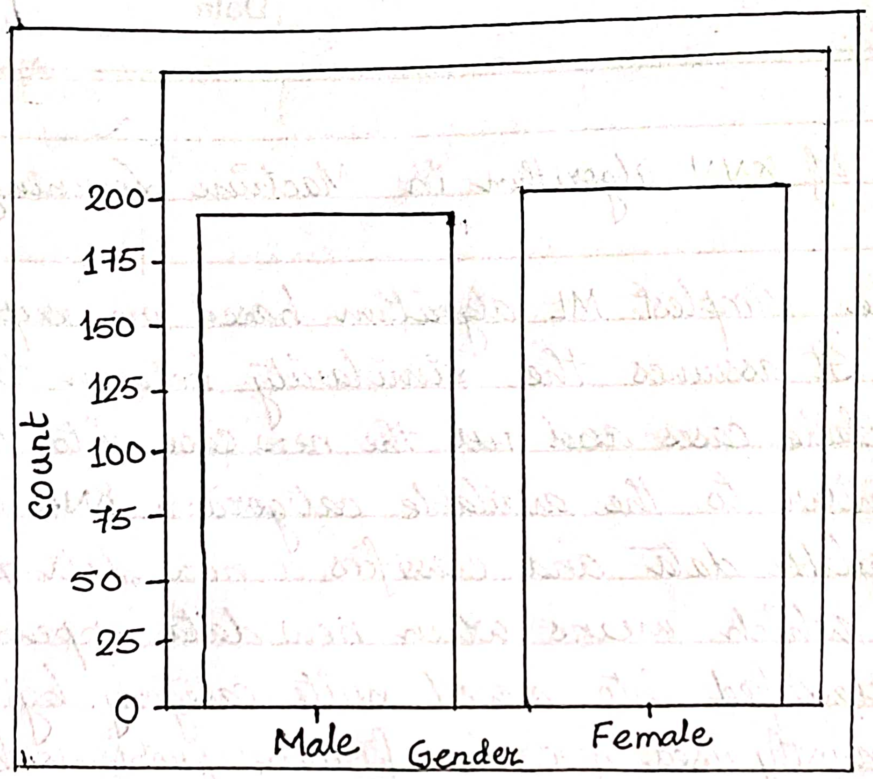
Expt, No.

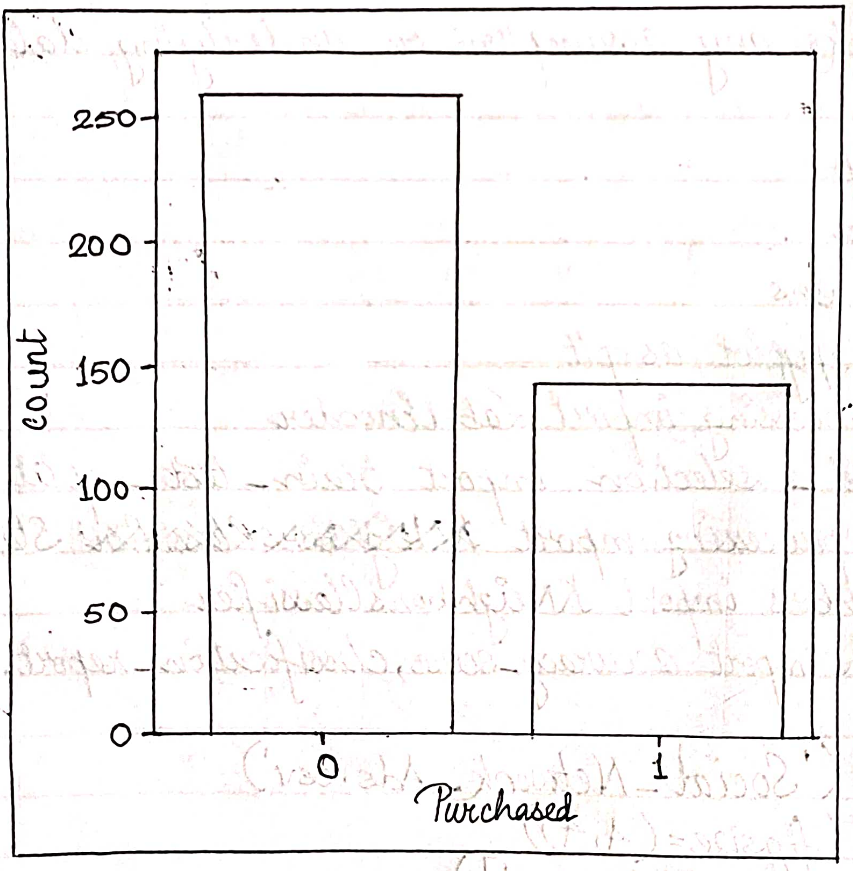
5) Implementation of KNN algorithm in Machine Learning.

'''KNN is one of the simplest ML algorithm based on supervised learning technique. It assumes the similarity between the new case or data and available cases and put the new case into the category that is most similar to the available categories. KNN algorithm stores all the available data and classifies a new data point based on the similarity which means when new data appears, then it can be easily classified into a well suite category by using KNN algorithm. It is mostly used for classification problems but can be used for regression problems too. KNN is a non-parametric algorithm, ie., it does not make any assumption on underlying data.'''

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import ~~KNeighborsClassifier~~ StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

df = pd.read_csv('Social_Network_Ads.csv')
ax = plt.subplots (figsize=(4,4))
ax= sns.countplot (x=df ['Gender'])
```


200
175
150
125
100
75
50
25
0
count
Male
Gender
Female


250
200
150
100
50
0
count
0
1
Purchased

Expt, No.

Page No. 06

Date

```
plt.show()
ax= plt.subplots(figsize=(4,4))
ax= sns.countplot(x=df['Purchased'])
plt.show()
X= df.iloc[:,[1,2,3]].values # Considering age, gender, salary as features
y=df.iloc[:,4].values
label_encoder = LabelEncoder()
x[:, 0] = label_encoder.fit_transform(x[:,0])
X_train, X_test, y_train, y_test = train_test_split(x, y, test=size=0.2,
                                                          random_state=42)
scaler= StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
k_value= 5 # This value can be adjusted based on preference
knn_classifier= KNeighborsClassifier(n_neighbors=k_value)
knn_classifier.fit(X_train, y_train)
y_pred = knn_classifier.predict(X_test)
accuracy= accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
classification_report_str = classification_report(y_test, y_pred)
print(f'KNN Accuracy: {accuracy}')
print(f'KNN Confusion Matrix:\n{conf_matrix}')
sns.set(rc={'figure.figsize':(6,3)})
sns.heatmap(confusion_matrix(y_test, y_pred), annot=True, fmt='d')
plt.xlabel('Predicted Labels')
plt.ylabel('Actual Labels')
```

Teacher's Signature :..........................

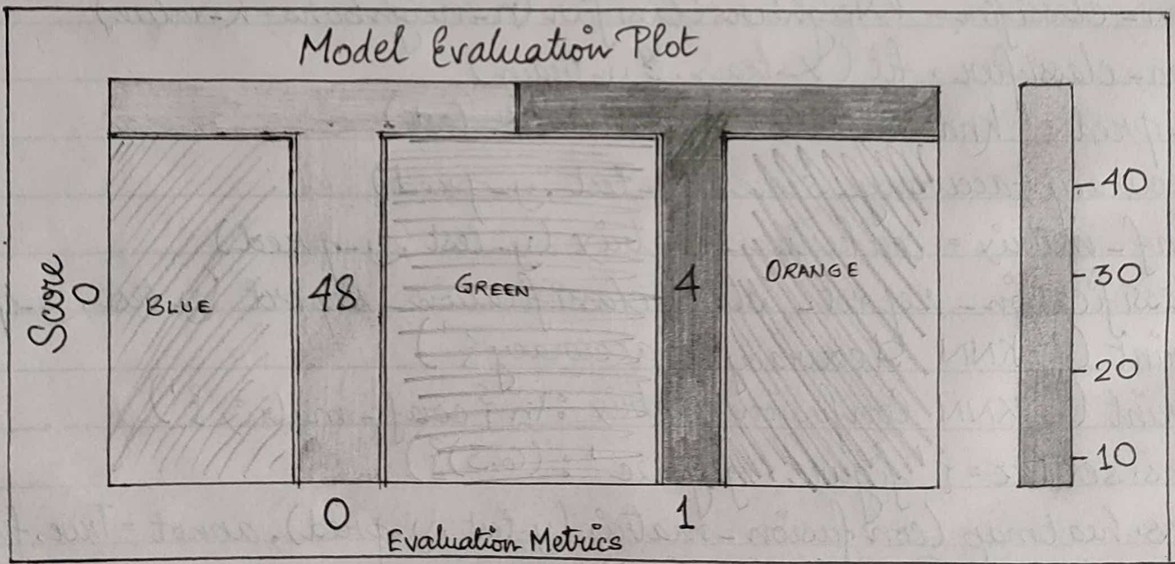
```
KNN Accuracy : 0.925
KNN Confusion Matrix :
[[48  4]
 [ 2 26]]
KNN Classification Report :

              precision    recall  f1-score   support

           0       0.96      0.92      0.94        52
           1       0.87      0.93      0.90        28

    accuracy                           0.93        80
   macro avg       0.91      0.93      0.92        80
weighted avg       0.93      0.93      0.93        80
```

The targeted audience is predicted not to purchase the product.



```
Cross-Validation Results:
Individual Accuracies : [0.675 0.8   0.775 0.9   0.8   0.75  0.925 0.85  0.75  0.8]
```

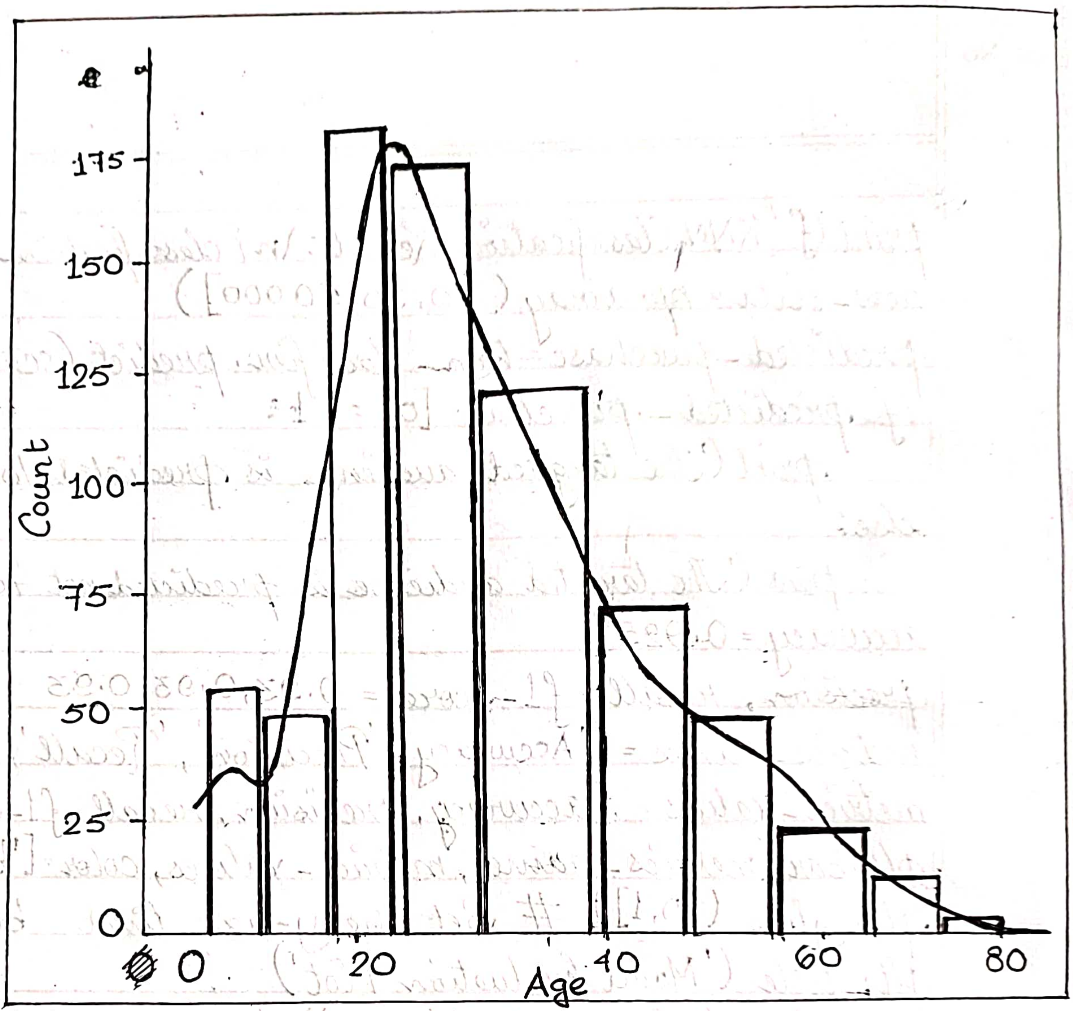
Expt, No.

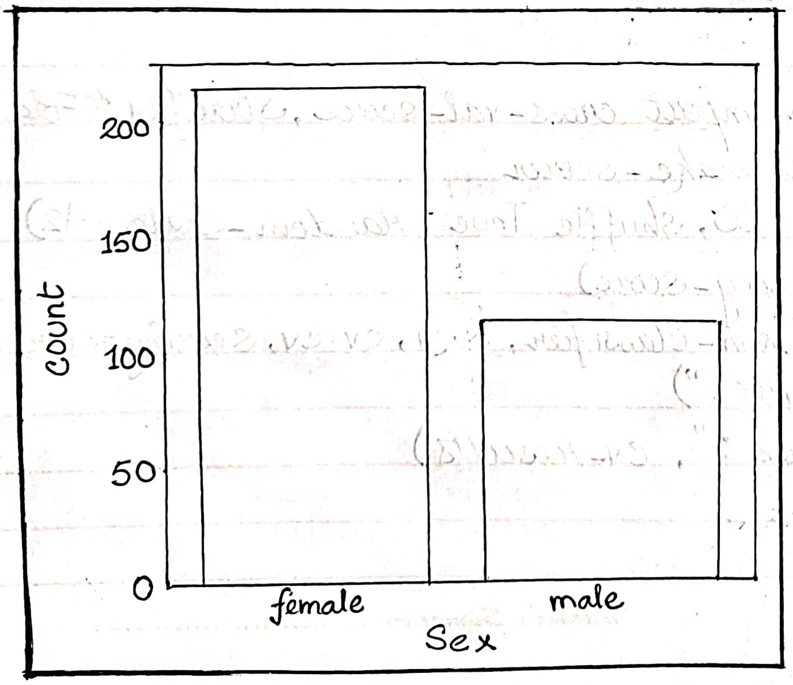
Date

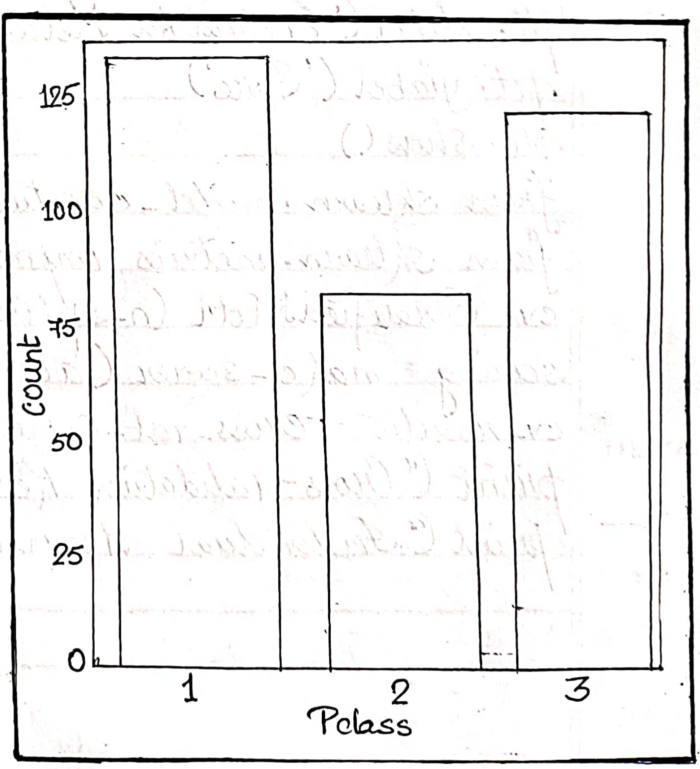
```
print(f'KNN Classification Report:\n{classification_report_str}')
new_data = np.array([0,30,50000])
predicted_purchase=knn_classifier.predict(scaler.transform(new_data))
if predicted_purchase[0] == 1:
    print('The targeted audience is predicted to purchase the product')
else:
    print('The targeted audience is predicted not to purchase the product')
accuracy = 0.925
precision, recall, f1_score = 0.93,0.93,0.93
metrics_names = ['Accuracy', 'Precision', 'Recall', 'F1-Score']
metrics_values = [accuracy, precision, recall, f1_score]
plt.bar(metrics_names, metrics_values, color=['blue','green','orange','red'])
plt.ylim([0,1]) # Set the y-axis limit between 0 & 1
plt.title('Model Evaluation Plot')
plt.xlabel('Evaluation Metrics')
plt.ylabel('Score')
plt.show()
from sklearn.model_selection import cross_val_score, StratifiedKFold
from sklearn.metrics import make_scorer
cv= StratifiedKFold(n_splits=10, shuffle=True, random_state=42)
scoring = make_scorer(accuracy_score)
cv_results = cross_val_score(knn_classifier, X, y, cv=cv, scoring=scoring)
print("Cross-Validation Results:")
print("Individual Accuracies:", cv_results)
```

```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64

177
```







Expt, No.

Page No.	
Date	

6)

```
import pandas as pd
import seaborn as sns
df= pd.read_csv('titanic.csv')

print(df.isnull().sum())
print (df['Age'].isnull().sum())

survived_df = df[df['Survived']==1]
dead-df = df[df['Survived']==0]
df1=df['Age'].dropna()
df1=df.select_dtypes(include=['int64','float64'])

sns.displot(data= df, x='Age', bins=10, kde= True, color='red')

sns.countplot (data= survived-df, x= survived-df['Sex'])

sns.countplot(data= survived-df, x= survived-df['Pclass'])
```

Teacher's Signature :..........................