EXP. NO: 1 **DATE:**

CLASSIFICATION USING SVM MODEL

AIM

To build a machine learning model using SVM in python to classify the "Iris.csv" dataset.

ALGORITHM

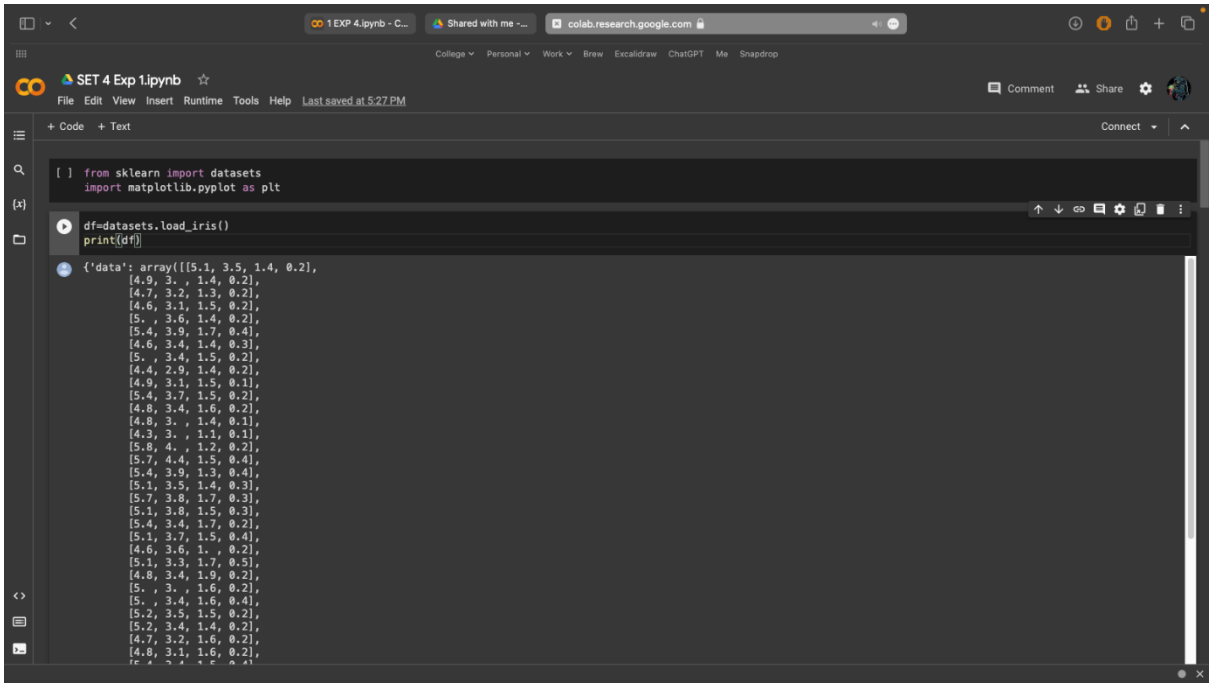
1. Import the required libraries including sklearn, pandas, and matplotlib.pyplot.
2. Load the iris dataset using datasets.load_iris().
3. Print the loaded dataset using print().
4. Print the feature names of the dataset using print().
5. Print the target names of the dataset using print().
6. Extract the data and target from the dataset using data=df.data and target=df.target.
7. Create a pandas dataframe df_data for the data with column names as feature names using pd.DataFrame(data, columns = df.feature_names).
8. Create a pandas dataframe target for the target variable using target=pd.DataFrame(target).
9. Split the dataset into training and testing sets using train_test_split(X, Y, test_size=0.2).
10. Fit the SVM model with SVC(gamma='auto') and predict the species for the given input vectors using svm.predict(). Also, print the accuracy score of the SVM model using accuracy_score().

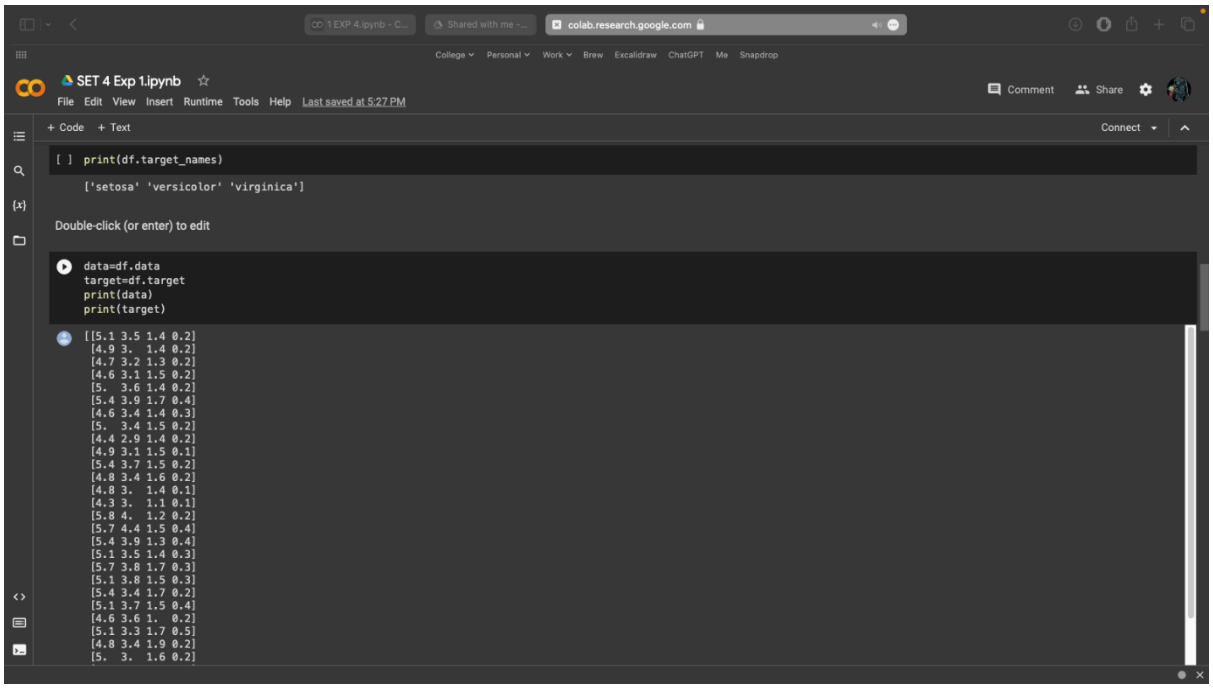
CODE SEGMENT

```
from sklearn import datasets
import matplotlib.pyplot as plt

# Loading dataset
df = datasets.load_iris()
print(df)

# Features
print(df.feature_names)

print(df.target_names)

data=df.data
target=df.target
print(data)
print(target)

import pandas as pd
```

```
df_data = pd.DataFrame(data, columns = df.feature_names)
target=pd.DataFrame(target)
df_data.head()

df_data.describe()

x=target[0].value_counts()
print(x)

import seaborn as sns
sns.barplot(x=x.index,y=x)

data = df.values
X = df_data
Y = target

# Split the data to train and test dataset.
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)

# Support vector machine algorithm
from sklearn.metrics import accuracy_score
from sklearn.svm import SVC
svm = SVC(gamma='auto')
svm.fit(X_train,y_train)
yhat = svm.predict(X_test)
print(y_test.T)
print(yhat)
svm_acc=accuracy_score(y_test, yhat)
print("Accuracy Score = ",accuracy_score(y_test, yhat))

from sklearn.metrics import classification_report
print(classification_report(y_test, yhat))

X_new = pd.DataFrame([[2, 2, 1, 0.2], [  4.9, 2.2, 2.8, 1.1 ], [  5.3, 2.5, 4.6, 1.9 ]])
#Prediction of the species from the input vector
prediction = svm.predict(X_new)
print("Prediction of Species:", prediction)
for i in prediction:
  print(df.target_names[i]
```

INPUT / OUTPUT





SET 4 Exp 1.ipynb

File Edit View Insert Runtime Tools Help Last saved at 5:27 PM

+ Code + Text

```
[ ] import pandas as pd
    df_data=pd.DataFrame(data,columns=df.feature_names)
    target=pd.DataFrame(target)
    df_data.head()
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

```
df_data.describe()
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333
std	0.828066	0.435866	1.765298	0.762238
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

```
[ ] x=target[0].value_counts()
```

SET 4 Exp 1.ipynb

File Edit View Insert Runtime Tools Help Last saved at 5:27 PM

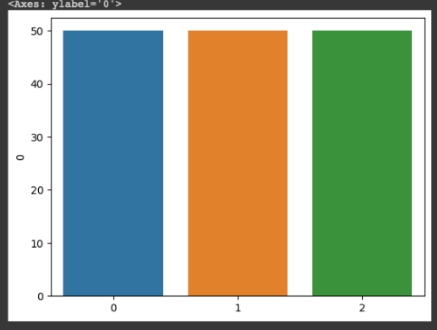
+ Code + Text

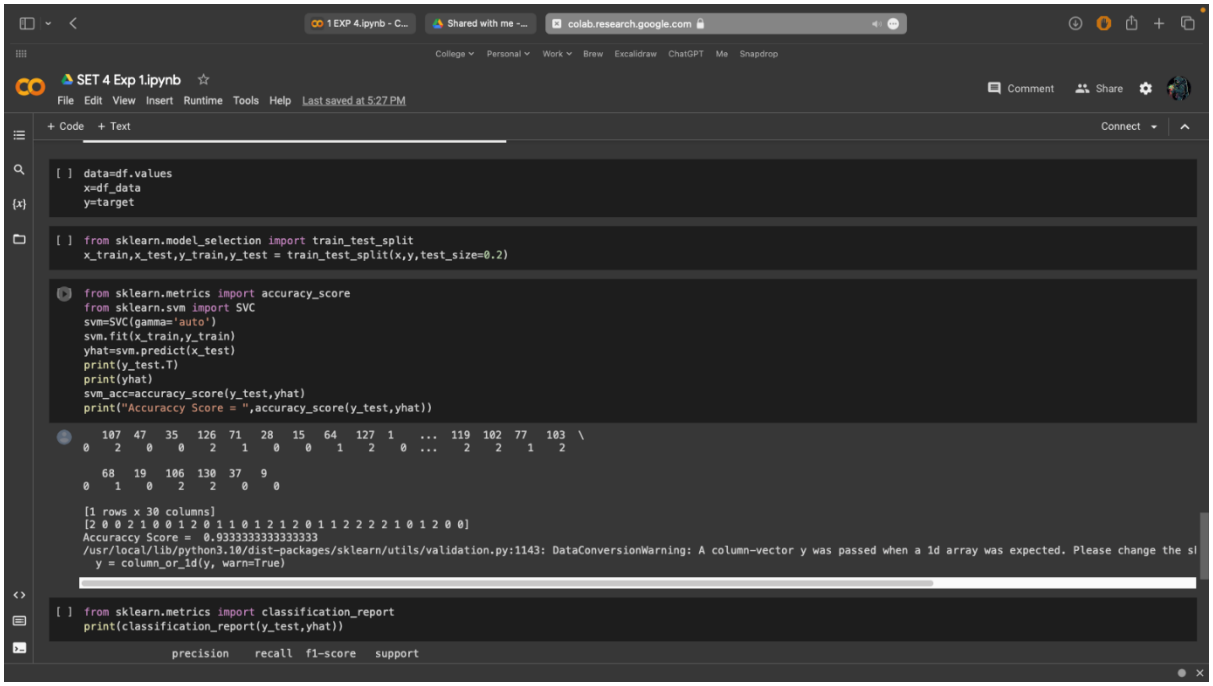
```
[ ] x=target[0].value_counts()
    print(x)
```

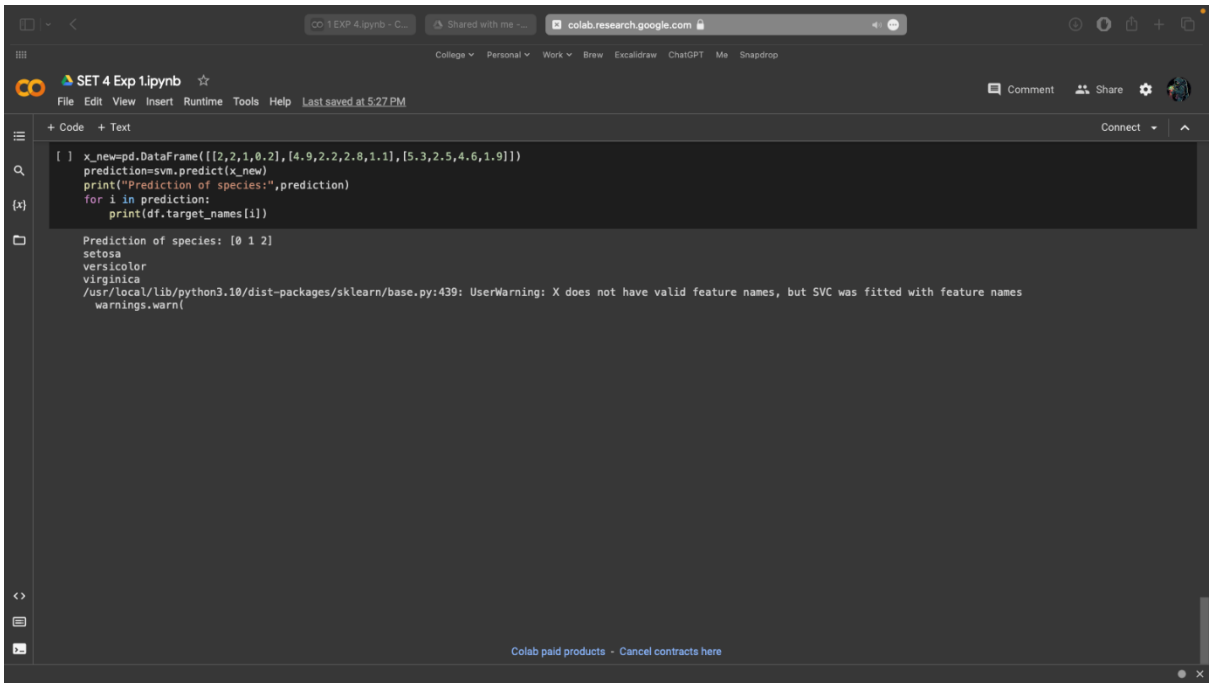
```
0    50
1    50
2    50
Name: 0, dtype: int64
```

```
[ ] import seaborn as sns
    sns.barplot(x=x.index,y=x)
```

```
<Axes: ylabel='0'>
```







RESULT

Thus, the SVM classification in python is successfully implemented and verified.

EXP. NO: 2 **DATE:**

CLASSIFICATION MODELS PERFORMANCE ANALYSIS

AIM

To analyze the performance of the different machine learning classification models using python.

ALGORITHM

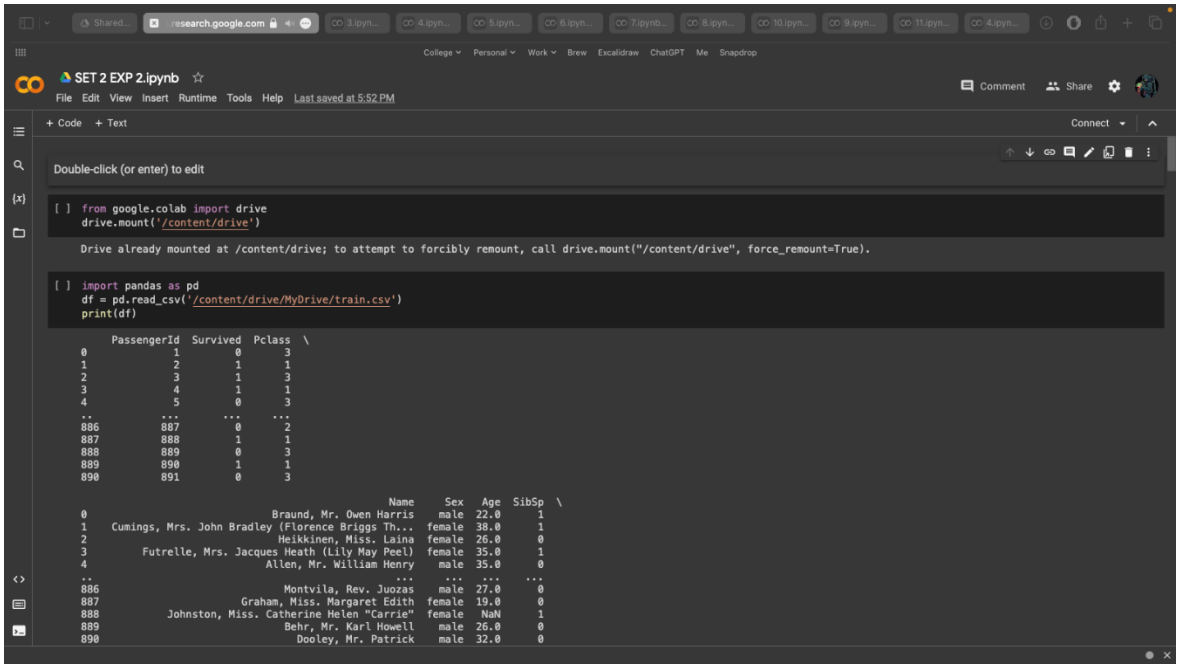
1. Import the required libraries
2. Load the dataset "train.csv" using Pandas and read the first few rows
3. Check for missing values in the loaded dataset and remove the rows with missing values using the "dropna()" function.
4. Create an empty list called "dummies" and create another list called "cols" with values "Pclass", "Sex", and "Embarked".
5. Use a for loop to iterate through each column in "cols" and use the "get_dummies()" function to create dummy variables for each column.
6. Concatenate the dummy variables generated in step 5 using the "concat()" function and assign to a variable called "tdummies".
7. Concatenate the original dataframe "df" with the tdummies dataframe generated in step 6 using the "concat()" function and drop the original columns 'Pclass', 'Sex', and 'Embarked' using the "drop()" function.
8. Interpolate the missing values in the 'Age' column using the "interpolate()" function.
9. Create variables "x" and "y" by storing the dataframe values and the "Survived" column values into respective variables.
10. Split the data into training and test sets using the "train_test_split()" function, and create a for loop that iterates through k values from 1 to 29. Inside the loop, train a KNN model using the "KNeighborsClassifier()" function with the current k value, fit the model on the training data, predict on the test data, store the accuracy score in a list called "acc", and print the accuracy score.

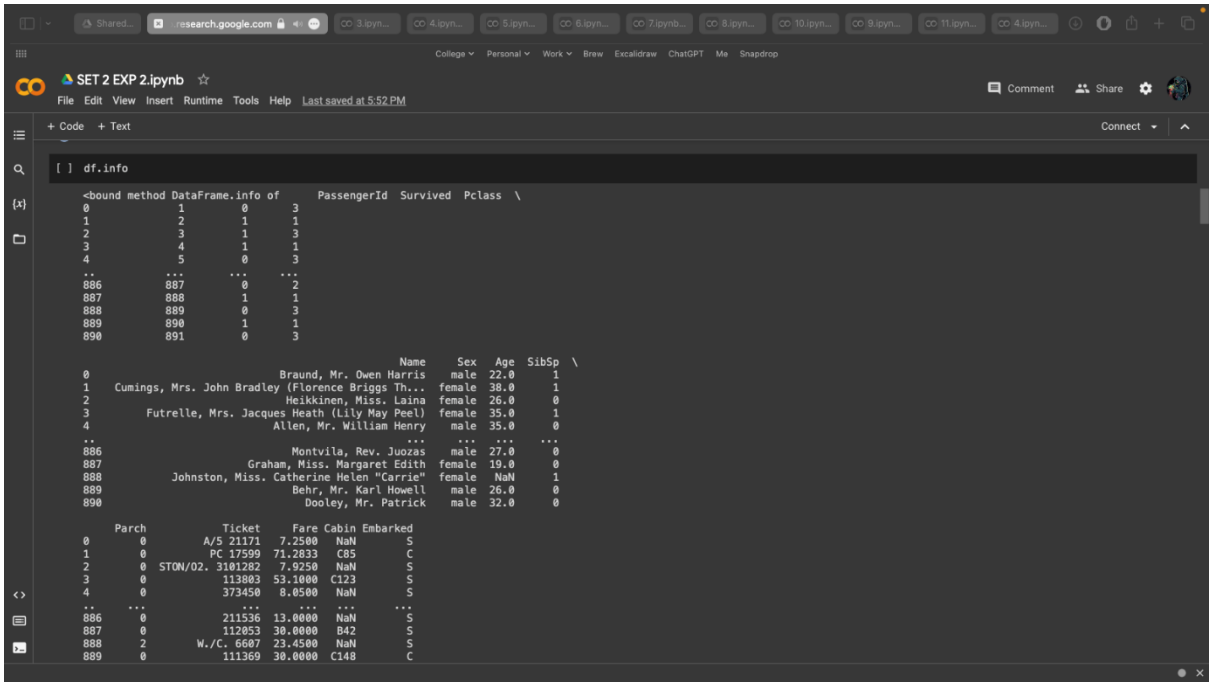
CODE SEGMENT

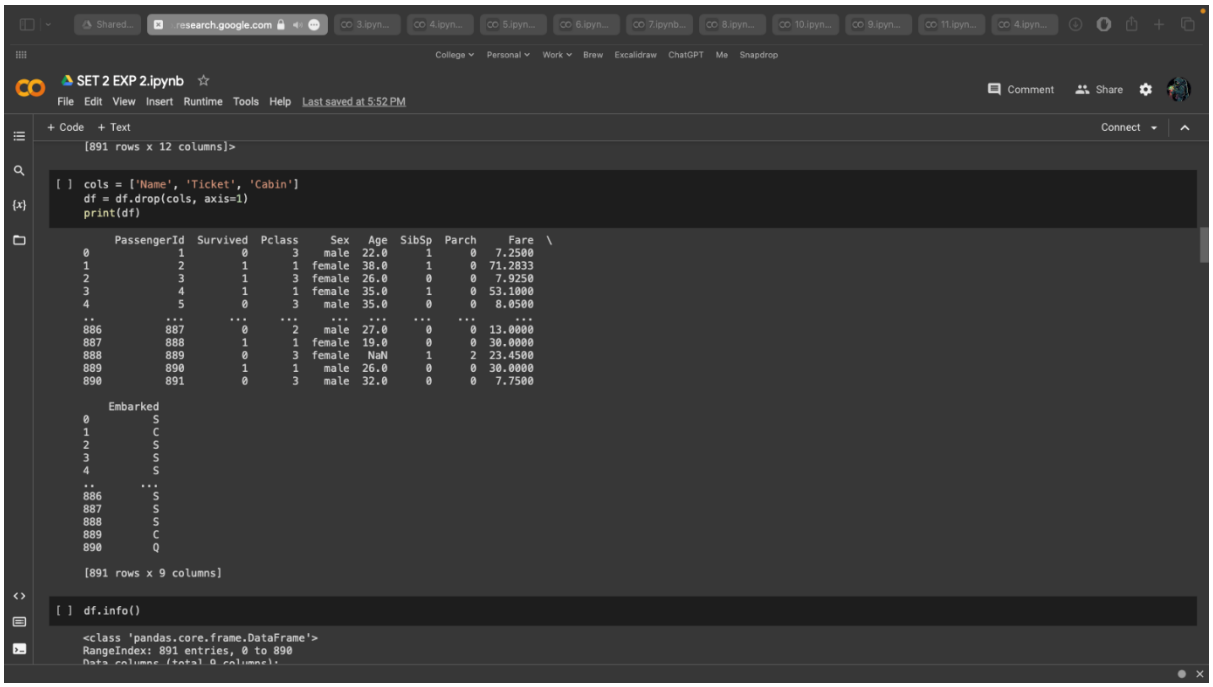
```
import pandas as pd
df = pd.read_csv('train.csv')
df.head()

df[df.isna().any(axis=1)]
df = df.dropna()
df.head()

dummies = []
cols = ['Pclass', 'Sex', 'Embarked']
```

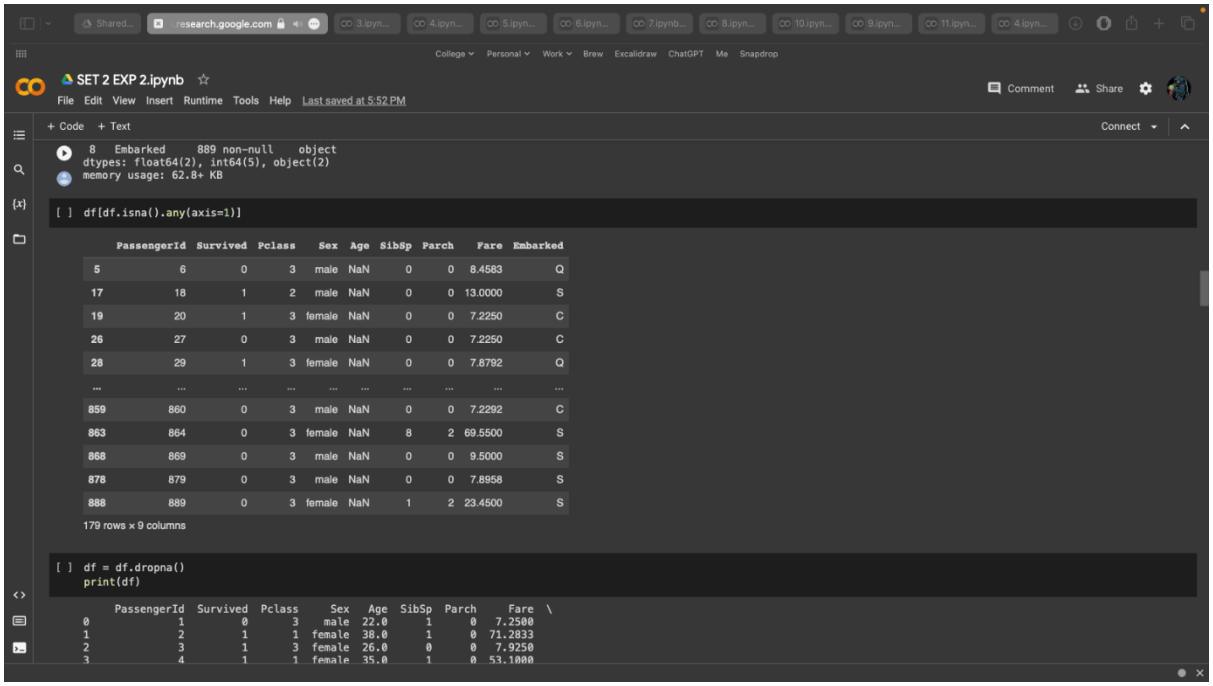
```
for col in cols:
  dummies.append(pd.get_dummies(df[col]))

tdummies = pd.concat(dummies, axis=1)
print(tdummies)

df = pd.concat((df,tdummies), axis=1)
df = df.drop(['Pclass', 'Sex', 'Embarked'], axis=1)
df['Age'] = df['Age'].interpolate()
df.head()

x = df.values
y = df['Survived'].values
print(x)
print(y)

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)
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
acc = []
for i in range(1,30):
    knn = KNeighborsClassifier(n_neighbors = i)
    knn.fit(x_train,y_train)
    yhat = knn.predict(x_test)
    acc.append(accuracy_score(y_test,yhat))
    print("For k = ",i," : ",accuracy_score(y_test,yhat))
```

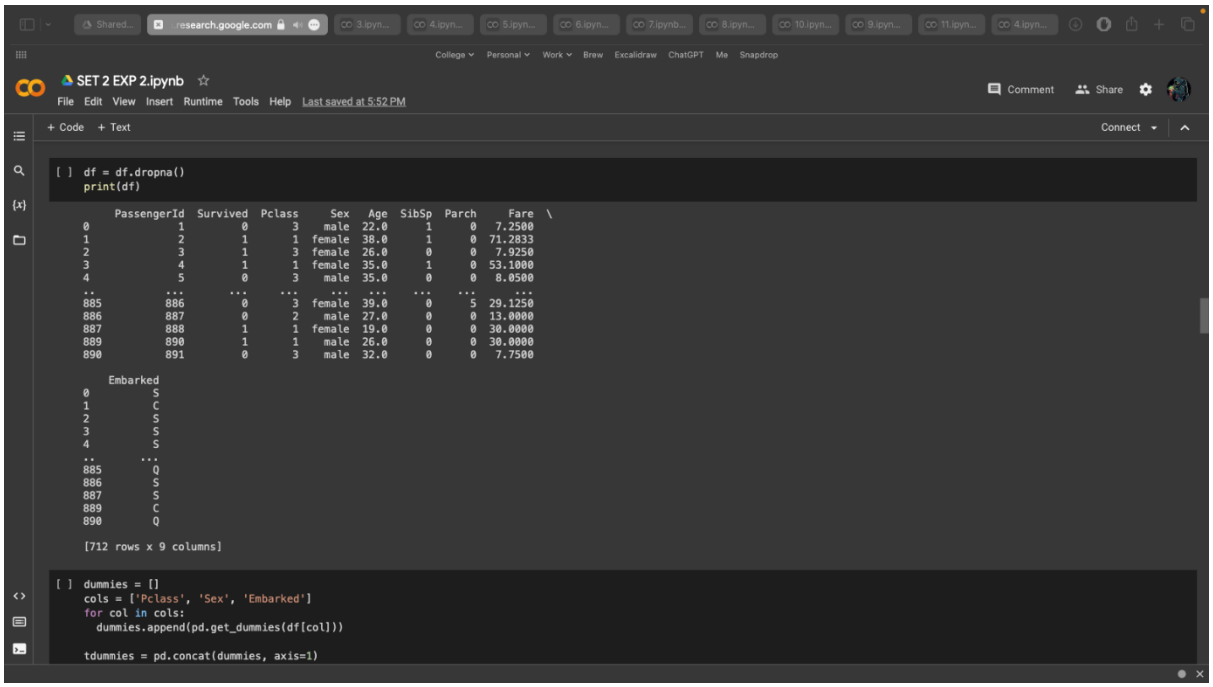
INPUT / OUTPUT

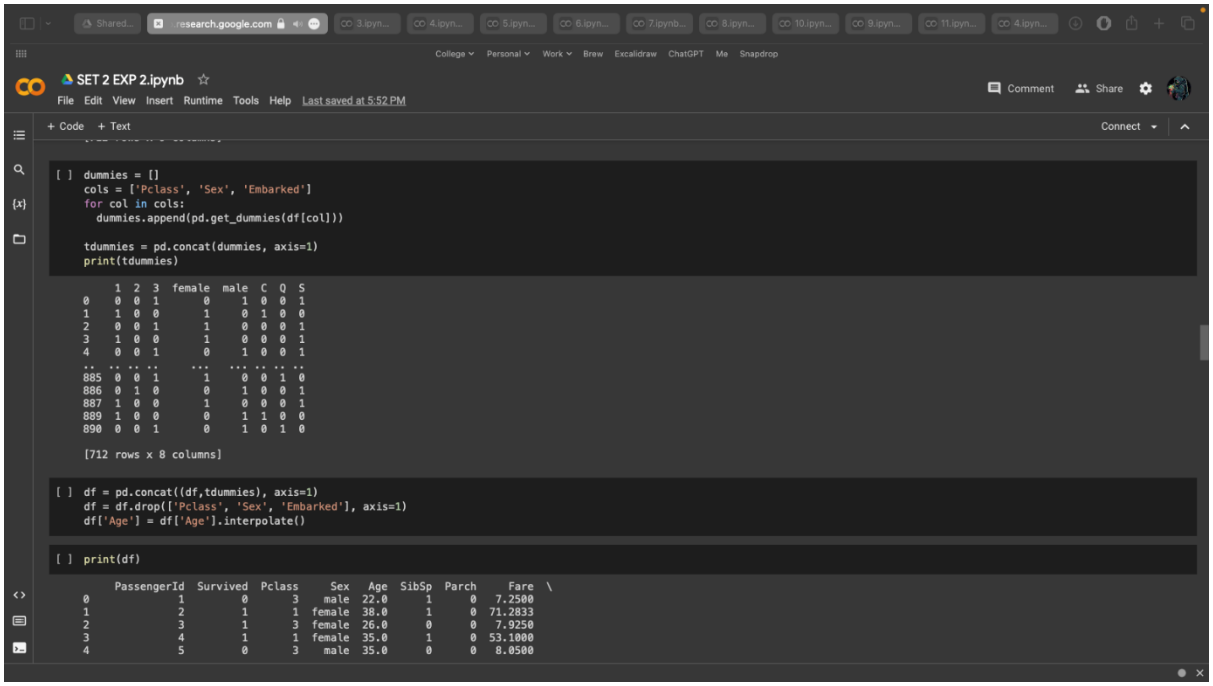


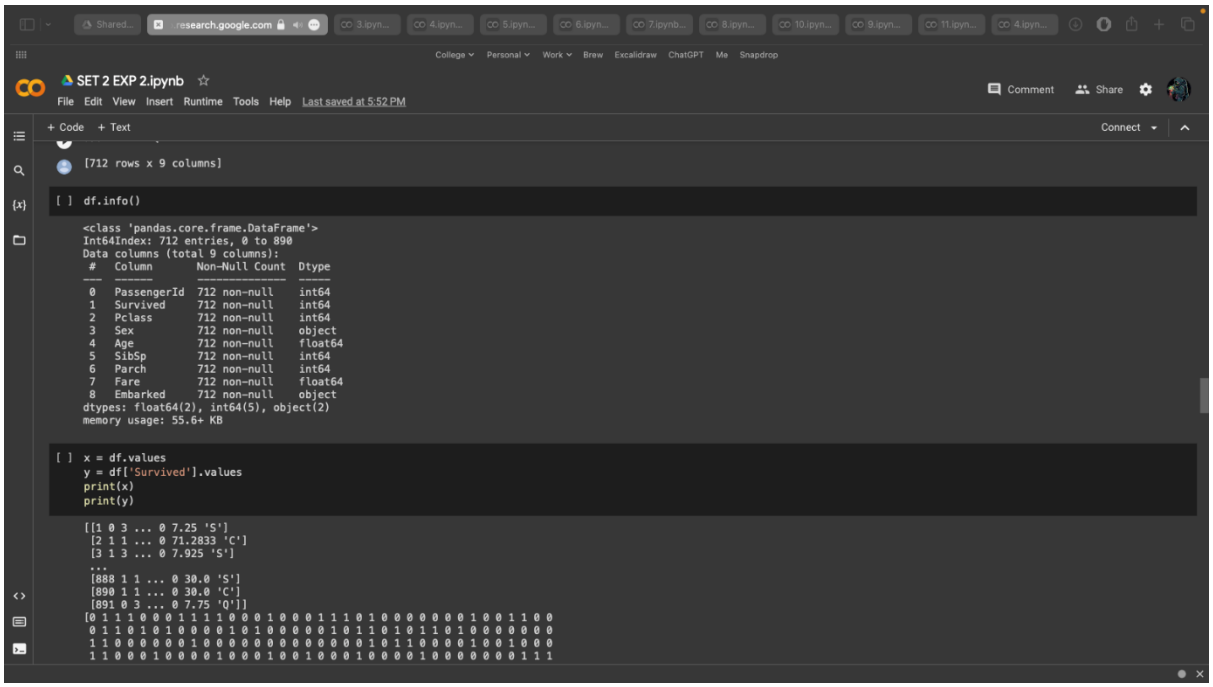

SET 2 EXP 2.ipynb
File Edit View Insert Runtime Tools Help Last saved at 5:52 PM
Comment
Share
Connect
+ Code + Text
[ ] df.info
<bound method DataFrame.info of PassengerId Survived Pclass \
0 1 0 3
1 2 1 1
2 3 1 3
3 4 1 1
4 5 0 3
.. ..
886 887 0 2
887 888 1 1
888 889 0 3
889 890 1 1
890 891 0 3
Name Sex Age SibSp \
0 Braund, Mr. Owen Harris male 22.0 1
1 Cumings, Mrs. John Bradley (Florence Briggs Th... female 38.0 1
2 Heikkinen, Miss. Laina female 26.0 0
3 Futrelle, Mrs. Jacques Heath (Lily May Peel) female 35.0 1
4 Allen, Mr. William Henry male 35.0 0
.. ..
886 Montvila, Rev. Juozas male 27.0 0
887 Graham, Miss. Margaret Edith female 19.0 0
888 Johnston, Miss. Catherine Helen "Carrie" female NaN 1
889 Behr, Mr. Karl Howell male 26.0 0
890 Dooley, Mr. Patrick male 32.0 0
Parch Ticket Fare Cabin Embarked
0 0 A/5 21171 7.2500 NaN S
1 0 PC 17599 71.2833 C85 C
2 0 STON/O2. 3101282 7.9250 NaN S
3 0 113803 53.1000 C123 S
4 0 373450 8.0500 NaN S
.. ..
886 0 211536 13.0000 NaN S
887 0 112053 30.0000 B42 S
888 2 W./C. 6607 23.4500 NaN S
889 0 111369 30.0000 C148 C


SET 2 EXP 2.ipynb
File Edit View Insert Runtime Tools Help Last saved at 5:52 PM
Comment
Share
Connect
+ Code + Text
[891 rows x 12 columns]>
[ ] cols = ['Name', 'Ticket', 'Cabin']
df = df.drop(cols, axis=1)
print(df)
PassengerId Survived Pclass Sex Age SibSp Parch Fare \
0 1 0 3 male 22.0 1 0 7.2500
1 2 1 1 female 38.0 1 0 71.2833
2 3 1 3 female 26.0 0 0 7.9250
3 4 1 1 female 35.0 1 0 53.1000
4 5 0 3 male 35.0 0 0 8.0500
.. ..
886 887 0 2 male 27.0 0 0 13.0000
887 888 1 1 female 19.0 0 0 30.0000
888 889 0 3 female NaN 1 2 23.4500
889 890 1 1 male 26.0 0 0 30.0000
890 891 0 3 male 32.0 0 0 7.7500
Embarked
0 S
1 C
2 S
3 S
4 S
.. ...
886 S
887 S
888 S
889 C
890 Q
[891 rows x 9 columns]
[ ] df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890


SET 2 EXP 2.ipynb
File Edit View Insert Runtime Tools Help Last saved at 5:52 PM
Comment
Share
Connect
+ Code + Text
8 Embarked 889 non-null object
dtypes: float64(2), int64(5), object(2)
memory usage: 62.8+ KB
[ ] df[df.isna().any(axis=1)]
PassengerId Survived Pclass Sex Age SibSp Parch Fare Embarked
5 6 0 3 male NaN 0 0 8.4583 Q
17 18 1 2 male NaN 0 0 13.0000 S
19 20 1 3 female NaN 0 0 7.2250 C
26 27 0 3 male NaN 0 0 7.2250 C
28 29 1 3 female NaN 0 0 7.8792 Q
859 860 0 3 male NaN 0 0 7.2292 C
863 864 0 3 female NaN 8 2 69.5500 S
868 869 0 3 male NaN 0 0 9.5000 S
878 879 0 3 male NaN 0 0 7.8958 S
888 889 0 3 female NaN 1 2 23.4500 S
179 rows x 9 columns
[ ] df = df.dropna()
print(df)
PassengerId Survived Pclass Sex Age SibSp Parch Fare \
0 1 0 3 male 22.0 1 0 7.2500
1 2 1 1 female 38.0 1 0 71.2833
2 3 1 3 female 26.0 0 0 7.9250
3 4 1 1 female 35.0 1 0 53.1000


SET 2 EXP 2.ipynb
File Edit View Insert Runtime Tools Help Last saved at 5:52 PM
Comment
Share
Connect
+ Code + Text
[ ] df = df.dropna()
print(df)
PassengerId Survived Pclass Sex Age SibSp Parch Fare \
0 1 0 3 male 22.0 1 0 7.2500
1 2 1 1 female 38.0 1 0 71.2833
2 3 1 3 female 26.0 0 0 7.9250
3 4 1 1 female 35.0 1 0 53.1000
4 5 0 3 male 35.0 0 0 8.0500
885 886 0 3 female 39.0 0 5 29.1250
886 887 0 2 male 27.0 0 0 13.0000
887 888 1 1 female 19.0 0 0 30.0000
889 890 1 1 male 26.0 0 0 30.0000
890 891 0 3 male 32.0 0 0 7.7500
Embarked
0 S
1 C
2 S
3 S
4 S
885 Q
886 S
887 S
889 C
890 Q
[712 rows x 9 columns]
[ ] dummies = []
cols = ['Pclass', 'Sex', 'Embarked']
for col in cols:
dummies.append(pd.get_dummies(df[col]))
tdummies = pd.concat(dummies, axis=1)


SET 2 EXP 2.ipynb
File Edit View Insert Runtime Tools Help Last saved at 5:52 PM
Comment
Share
Connect
+ Code + Text
dummies = []
cols = ['Pclass', 'Sex', 'Embarked']
for col in cols:
dummies.append(pd.get_dummies(df[col]))
tdummies = pd.concat(dummies, axis=1)
print(tdummies)
1 2 3 female male C Q S
0 0 0 1 0 1 0 0 1
1 1 0 0 1 0 1 0 0
2 0 0 1 1 0 0 0 1
3 1 0 0 1 0 0 0 1
4 0 0 1 0 1 0 0 1
885 0 0 1 1 0 0 1 0
886 0 1 0 0 1 0 0 1
887 1 0 0 1 0 0 0 1
889 1 0 0 0 1 1 0 0
890 0 0 1 0 1 0 1 0
[712 rows x 8 columns]
df = pd.concat((df,tdummies), axis=1)
df = df.drop(['Pclass', 'Sex', 'Embarked'], axis=1)
df['Age'] = df['Age'].interpolate()
print(df)
PassengerId Survived Pclass Sex Age SibSp Parch Fare \
0 1 0 3 male 22.0 1 0 7.2500
1 2 1 1 female 38.0 1 0 71.2833
2 3 1 3 female 26.0 0 0 7.9250
3 4 1 1 female 35.0 1 0 53.1000
4 5 0 3 male 35.0 0 0 8.0500


SET 2 EXP 2.ipynb
File Edit View Insert Runtime Tools Help Last saved at 5:52 PM
Comment
Share
Connect
+ Code + Text
[712 rows x 9 columns]
df.info()
<class 'pandas.core.frame.DataFrame'>
Int64Index: 712 entries, 0 to 890
Data columns (total 9 columns):
Column Non-Null Count Dtype
0 PassengerId 712 non-null int64
1 Survived 712 non-null int64
2 Pclass 712 non-null int64
3 Sex 712 non-null object
4 Age 712 non-null float64
5 SibSp 712 non-null int64
6 Parch 712 non-null int64
7 Fare 712 non-null float64
8 Embarked 712 non-null object
dtypes: float64(2), int64(5), object(2)
memory usage: 55.6+ KB
x = df.values
y = df['Survived'].values
print(x)
print(y)
[[1 0 3 ... 0 7.25 'S']
[2 1 1 ... 0 71.2833 'C']
[3 1 3 ... 0 7.925 'S']
...
[888 1 1 ... 0 30.0 'S']
[890 1 1 ... 0 30.0 'C']
[891 0 3 ... 0 7.75 'Q']]
[0 1 1 1 0 0 0 1 1 1 1 0 0 0 1 0 0 0 1 1 1 0 1 0 0 0 0 0 0 0 1 0 0 1 1 0 0
0 1 1 0 1 0 1 0 1 0 0 0 0 1 0 1 0 0 0 0 0 1 0 1 1 0 1 0 1 1 0 1 0 0 0 0 0 0 0
1 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 0 0 0 0 1 0 0 1 0 0 0
1 1 0 0 0 1 0 0 0 0 1 0 0 0 1 0 0 1 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 1 1 1

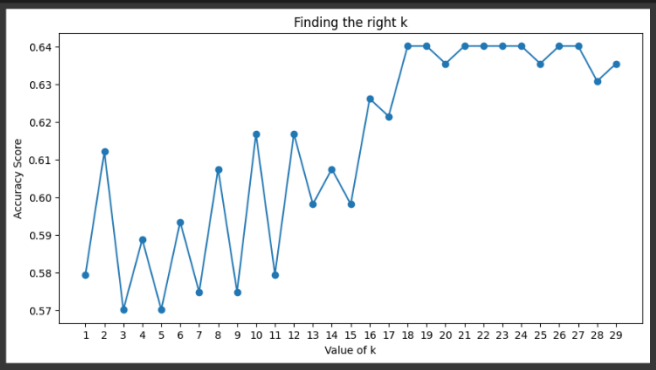
SET 2 EXP 2.ipynb

File Edit View Insert Runtime Tools Help Last saved at 5:52 PM

Comment Share Connect

+ Code + Text

```
[ ] import matplotlib.pyplot as plt
    plt.figure(figsize=(10,5))
    plt.plot(range(1,30),acc, marker = "o")
    plt.xlabel("Value of k")
    plt.ylabel("Accuracy Score")
    plt.title("Finding the right k")
    plt.xticks(range(1,30))
    plt.show()
```




```
[ ] KNN = KNeighborsClassifier(n_neighbors = 9)
```

SET 2 EXP 2.ipynb

File Edit View Insert Runtime Tools Help Last saved at 5:52 PM

Comment Share Connect

+ Code + Text

```
[ ] KNN = KNeighborsClassifier(n_neighbors = 9)
    print(x_test)
    KNN.fit(x,y)
    yhat = KNN.predict(x_test)
    knn_acc = accuracy_score(y_test,yhat)
    print("Accuracy Score = ",accuracy_score(y_test,yhat))
```

```
[[424.     0.    28.   ...   0.     0.     1.  ]
 [179.     0.    30.   ...   0.     0.     1.  ]
 [306.     1.     0.92 ...   0.     0.     1.  ]
 ...
 [672.     0.    31.   ...   0.     0.     1.  ]
 [581.     1.    25.   ...   0.     0.     1.  ]
 [337.     0.    29.   ...   0.     0.     1.  ]]
Accuracy Score =  0.6495327102803738
```

```
[ ] from sklearn.tree import DecisionTreeClassifier
    depth = [];
    for i in range(1,8):
      clf_tree = DecisionTreeClassifier(criterion="entropy",random_state = 100, max_depth = i)
      clf_tree.fit(x_train,y_train)
      yhat = clf_tree.predict(x_test)
      depth.append(accuracy_score(y_test,yhat))
      print("For max depth = ",i, ": ",accuracy_score(y_test,yhat))
    DTC_acc=accuracy_score(y_test,yhat)
```

```
For max depth =  1 :  1.0
For max depth =  2 :  1.0
For max depth =  3 :  1.0
For max depth =  4 :  1.0
For max depth =  5 :  1.0
For max depth =  6 :  1.0
For max depth =  7 :  1.0
```

```
[ ] plt.figure(figsize=(8,6))
    plt.plot(range(1,8), depth, color="red", marker = "o")
```

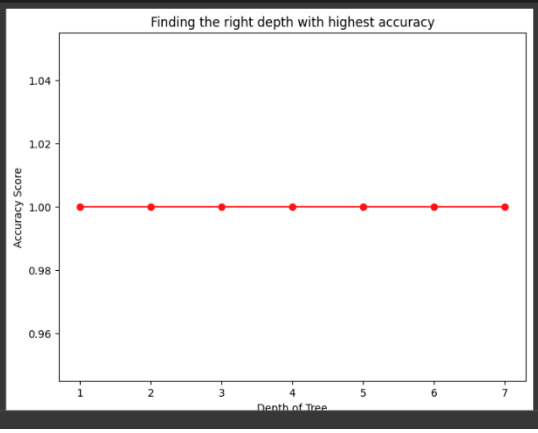
SET 2 EXP 2.ipynb

File Edit View Insert Runtime Tools Help Last saved at 5:52 PM

Comment Share Connect

+ Code + Text

```
[ ] plt.figure(figsize=(8,6))
    plt.plot(range(1,8), depth, color="red", marker = "o")
    plt.xlabel("Depth of Tree")
    plt.ylabel("Accuracy Score")
    plt.title("Finding the right depth with highest accuracy")
    plt.xticks(range(1,8))
    plt.show()
```



SET 2 EXP 2.ipynb

File Edit View Insert Runtime Tools Help Last saved at 5:52 PM

Comment Share Connect

+ Code + Text

```
[ ] svm = SVC(gamma='auto')
    svm.fit(x_train,y_train)
    yhat=svm.predict(x_test)
    print("Accuracy Score = ",accuracy_score(y_test, yhat))
    svm_acc = accuracy_score(y_test, yhat)
```

```
Accuracy Score =  0.5841121495327103
```

```
[ ] from sklearn.naive_bayes import GaussianNB
    NB = GaussianNB()
    NB.fit(x_train,y_train)
    yhat = NB.predict(x_test)
    print("Accuracy for training data: ",accuracy_score(y_test,yhat))
    NB_acc = accuracy_score(y_test, yhat)
```

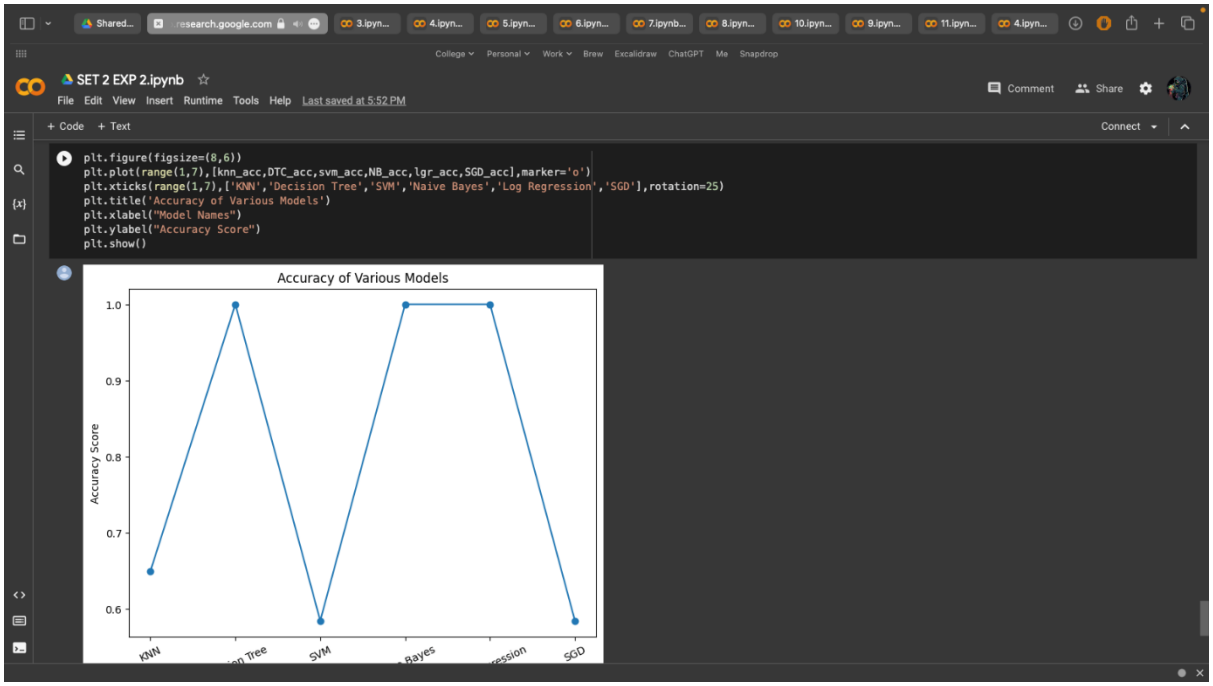
```
Accuracy for training data:  1.0
```

```
from sklearn.linear_model import LogisticRegression
regr = LogisticRegression(solver='liblinear', random_state=1)
regr.fit(x_train,y_train)
yhat=regr.predict(x_test)
print("Accuracy for training data: ",accuracy_score(y_test,yhat))
lgr_acc = accuracy_score(y_test, yhat)
```

```
Accuracy for training data:  1.0
```

```
[ ] from sklearn.linear_model import SGDClassifier
    SGD = SGDClassifier(loss="squared_error", penalty="l2", max_iter=4500, tol=0.0001, random_state=1)
    SGD.fit(x_train, y_train)
    yhat=SGD.predict(x_test)
    print(accuracy_score(y_test,yhat))
    SGD_acc=accuracy_score(y_test,yhat)
```

```
0.5841121495327103
```



RESULT

Thus, the performance of the different classification models using python have been analyzed and executed successfully.

EXP. NO: 3 **DATE:**

NAÏVE BAYE'S CLASSIFIER

AIM

To perform machine learning - classification using Naive Baye's Classifier in python.

ALGORITHM

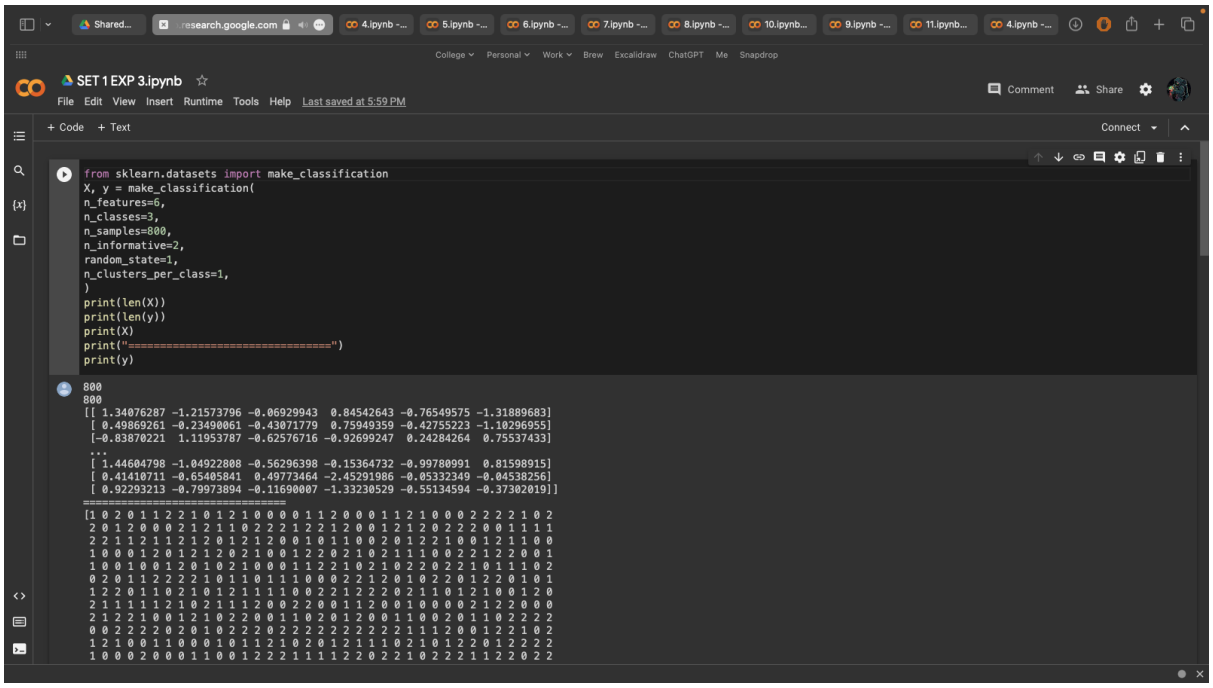
1. Import the necessary libraries for generating the classification dataset and for building and evaluating the model.
2. Use the make_classification function from sklearn.datasets to create a classification dataset with 800 samples, 6 features, 3 classes, and 2 informative features.
3. Print the length of the feature matrix X and target array y to verify the size of the dataset.
4. Print the feature matrix X and target array y to view the dataset.
5. Use matplotlib to create a scatter plot of the feature matrix X with the target array y as the colors.
6. Use train_test_split from sklearn.model_selection to split the dataset into training and testing sets with a 33% test size and a random state of 125.
7. Use GaussianNB from sklearn.naive_bayes to build a Gaussian Naive Bayes classifier model.
8. Fit the model to the training data using the fit method.
9. Use the predict method to make a prediction on a single test instance and print the actual value and predicted value.
10. Use accuracy_score, f1_score, and confusion_matrix from sklearn.metrics to evaluate the model's performance on the test set and print the accuracy and F1 score, and plot the confusion matrix using ConfusionMatrixDisplay.

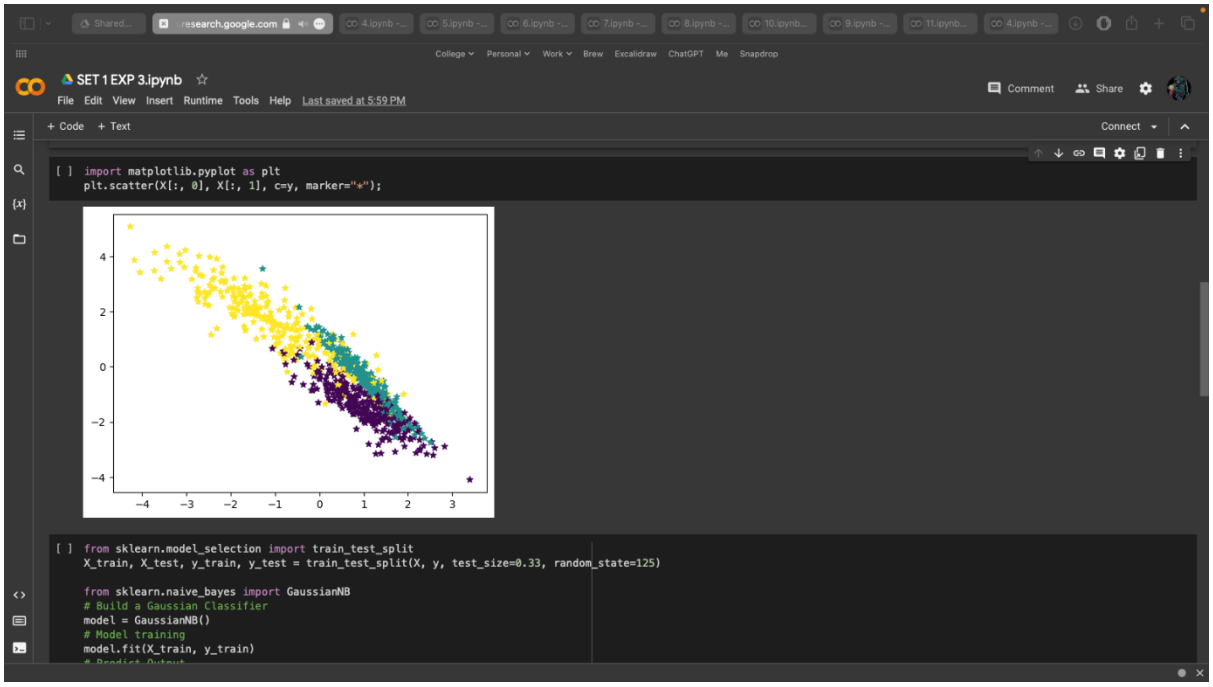
CODE SEGMENT

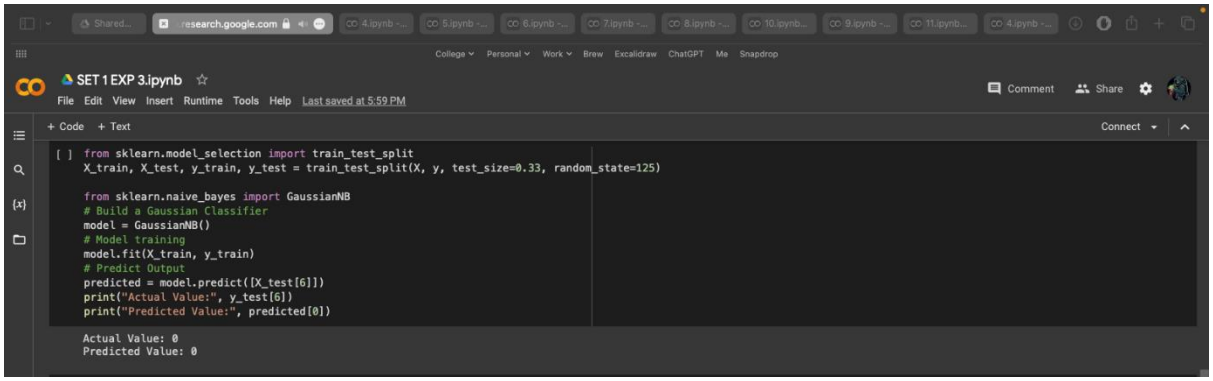
```
from sklearn.datasets import make_classification
X, y = make_classification(
n_features=6,
n_classes=3,
n_samples=800,
n_informative=2,
random_state=1,
n_clusters_per_class=1,
)
print(len(X))
print(len(y))
```

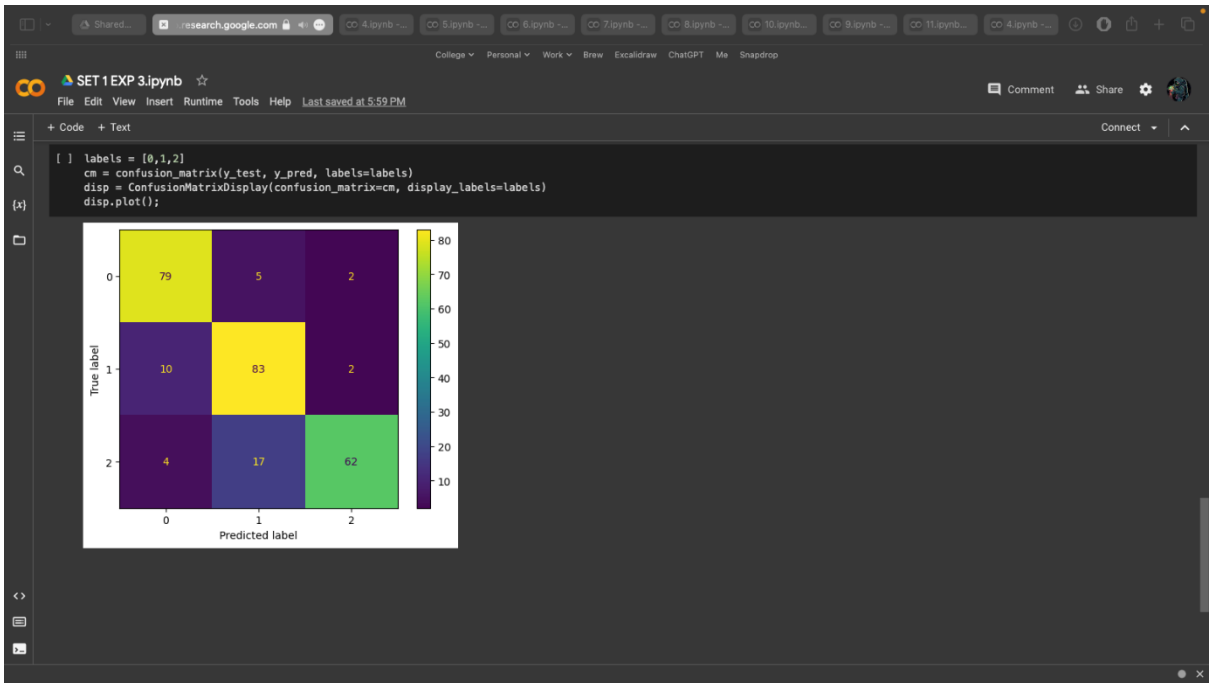
```
print(X)
print("================================")
print(y)

import matplotlib.pyplot as plt
plt.scatter(X[:, 0], X[:, 1], c=y, marker="*");

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=125)

from sklearn.naive_bayes import GaussianNB
# Build a Gaussian Classifier
model = GaussianNB()
# Model training
model.fit(X_train, y_train)
# Predict Output
predicted = model.predict([X_test[6]])
print("Actual Value:", y_test[6])
print("Predicted Value:", predicted[0])

from sklearn.metrics import (
accuracy_score,
confusion_matrix,
ConfusionMatrixDisplay,
f1_score,
)
y_pred = model.predict(X_test)
accuray = accuracy_score(y_pred, y_test)
f1 = f1_score(y_pred, y_test, average="weighted")
print("Accuracy:", accuray)
print("F1 Score:", f1)

labels = [0,1,2]
cm = confusion_matrix(y_test, y_pred, labels=labels)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=labels)
disp.plot();
```

INPUT / OUTPUT









RESULT

Thus, the Naive Baye's Classifier to perform machine learning classification in python has been implemented and executed successfully.

EXP. NO: 4 **DATE:**

BAYESIAN NETWORK CLASSIFICATION

AIM

To perform machine learning - classification using Bayesian Network Classifier in python.

ALGORITHM

1. Import the drive from google.colab package
2. Mount the drive using drive.mount('/content/drive')
3. Import pandas package as pd
4. Read the csv file 'Buy_Computer.csv' using the pandas read_csv method and store it in a variable df.
5. Print the DataFrame df using the print() method
6. Calculate the number of instances where Buy_Computer is 'yes' and 'no' using value_counts() method and store it in variables x0y and x0n respectively.
7. Calculate the prior probability of Buy_Computer being 'yes' and 'no' by dividing x0y by x0y+x0n and x0n by x0y+x0n respectively and store it in variables pp_target_yes and pp_target_no.
8. Count the frequency of each unique value in the age column using the value_counts() method and store it in variable x1.
9. Print the total number of age values using sum() method on variable x1.
10. Print the values in variable x1 using print() method.
11. Create empty lists age_yes and age_no.
12. For each unique age value, count the frequency of 'yes' for the Buy_Computer column and store it in variables x11y, x12y and x13y for 'youth', 'senior' and 'middle_age' respectively. Append these values to the age_yes list.
13. For each unique age value, count the frequency of 'no' for the Buy_Computer column and store it in variables x11n, x12n and x13n for 'youth', 'senior' and 'middle_age' respectively. Append these values to the age_no list.
14. For each value in x1 list, calculate the conditional probability of Buy_Computer being 'yes' given the corresponding age group by dividing the frequency of 'yes' in age_yes list by the frequency of the age group in x1 list and append the result to the pp_age_yes list. Similarly, calculate the conditional probability of Buy_Computer being 'no' given the corresponding age group and append the result to the pp_age_no list. If the frequency of a given age group is zero, append 0 to the pp_age_yes and pp_age_no lists.
15. Repeat steps 8 to 14 for income, student, and credit_rating columns. Store the conditional probabilities in pp_income_yes, pp_income_no, pp_stu_yes, pp_stu_no, pp_credit_rating_yes, and pp_credit_rating_no lists.

CODE SEGMENT

```
from google.colab import drive
drive.mount('/content/drive')
```

```
import pandas as pd
df = pd.read_csv('Buy_Computer.csv')
print(df)

x0y=df['Buy_Computer'].value_counts()['yes']
x0n=df['Buy_Computer'].value_counts()['no']
pp_target_yes=x0y/(x0y+x0n)
pp_target_no=x0n/(x0y+x0n)
print(x0y,x0n)
print(pp_target_yes,pp_target_no)

x1=df['age'].value_counts()
age_order=['youth','senior','middle_age']
print(sum(x1))
print(x1)
age_yes=[]
age_no=[]
pp_age_yes=[]
pp_age_no=[]
x11y=df[(df['age']=='youth') & (df['Buy_Computer']=='yes')]['age'].count()
x12y=df[(df['age']=='senior') & (df['Buy_Computer']=='yes')]['age'].count()
x13y=df[(df['age']=='middle_age') & (df['Buy_Computer']=='yes')]['age'].count()

age_yes.append(x11y)
age_yes.append(x12y)
age_yes.append(x13y)

x11n=df[(df['age']=='youth') & (df['Buy_Computer']=='no')]['age'].count()
x12n=df[(df['age']=='senior') & (df['Buy_Computer']=='no')]['age'].count()
x13n=df[(df['age']=='middle_age') & (df['Buy_Computer']=='no')]['age'].count()


age_no.append(x11n)
age_no.append(x12n)
age_no.append(x13n)

print(age_yes,age_no)

for i in range(len(x1)):
    if (age_yes[i]>0):
        pp_age_yes.append(age_yes[i]/x1[i])
    else:
        pp_age_yes.append(0)
for i in range(len(x1)):
    if (age_no[i]>0):
        pp_age_no.append(age_no[i]/x1[i])
```

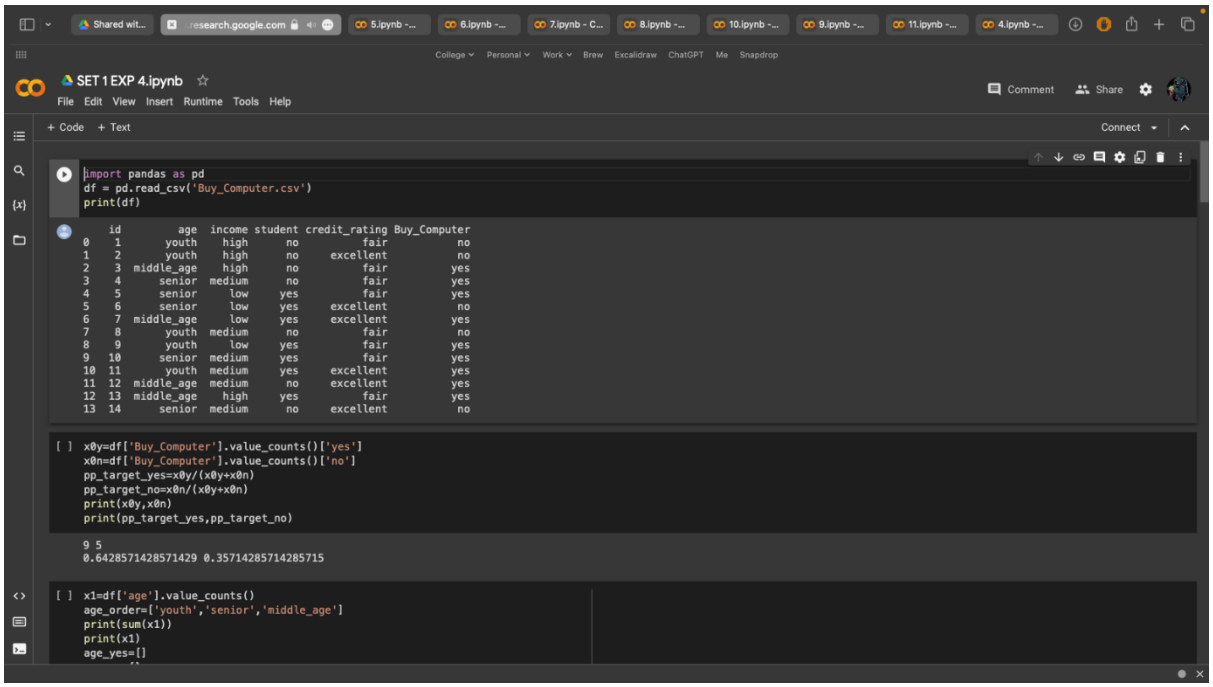
```
    else:
        pp_age_no.append(0)

print(pp_age_yes)
print(pp_age_no)

x2=df['income'].value_counts()
print(sum(x2))
print(x2)

income_yes=[]
income_no=[]
pp_income_yes=[]
pp_income_no=[]
x21y=df[(df['income']=='medium') & (df['Buy_Computer']=='yes')]['income'].count()
x22y=df[(df['income']=='high') & (df['Buy_Computer']=='yes')]['income'].count()
x23y=df[(df['income']=='low') & (df['Buy_Computer']=='yes')]['income'].count()
income_yes.append(x21y)
income_yes.append(x22y)
income_yes.append(x23y)


x21n=df[(df['income']=='medium') & (df['Buy_Computer']=='no')]['income'].count()
x22n=df[(df['income']=='high') & (df['Buy_Computer']=='no')]['income'].count()
x23n=df[(df['income']=='low') & (df['Buy_Computer']=='no')]['income'].count()
income_no.append(x21n)
income_no.append(x22n)
income_no.append(x23n)

print(income_yes,income_no)

for i in range(len(x2)):
  if (income_yes[i]>0):
      pp_income_yes.append(income_yes[i]/x2[i])
  else:
      pp_income_yes.append(0)
for i in range(len(x2)):
  if (income_no[i]>0):
    pp_income_no.append(income_no[i]/x2[i])
  else:
    pp_income_no.append(0)
print('Prior prob - high   Medium    Low')
print(pp_income_yes)
print(pp_income_no)

x3=df['student'].value_counts()
```

```
print(sum(x3))
print(x3)
stu_yes=[]
stu_no=[]
pp_stu_yes=[]
pp_stu_no=[]

x31y=df[(df['student']=='no') & (df['Buy_Computer']=='yes')]['student'].count()
x32y=df[(df['student']=='yes') & (df['Buy_Computer']=='yes')]['student'].count()


stu_yes.append(x31y)
stu_yes.append(x32y)

x31n=df[(df['student']=='no') & (df['Buy_Computer']=='no')]['student'].count()
x32n=df[(df['student']=='yes') & (df['Buy_Computer']=='no')]['student'].count()
stu_no.append(x31n)
stu_no.append(x32n)

print(stu_yes,stu_no)

for i in range(len(x3)):
  if (stu_yes[i]>0):
      pp_stu_yes.append(stu_yes[i]/x3[i])
  else:
      pp_stu_yes.append(0)
for i in range(len(x3)):
  if (stu_no[i]>0):
    pp_stu_no.append(stu_no[i]/x3[i])
  else:
    pp_stu_no.append(0)

print(pp_stu_yes)
print(pp_stu_no)

x4=df['credit_rating'].value_counts()
print(sum(x4))
print(x4)
credit_rating_yes=[]
credit_rating_no=[]
pp_credit_rating_yes=[]
pp_credit_rating_no=[]

x41y=df[(df['credit_rating']=='fair') & (df['Buy_Computer']=='yes')]['credit_rating'].count()
x42y=df[(df['credit_rating']=='excellent') &
(df['Buy_Computer']=='yes')]['credit_rating'].count()
```

```
credit_rating_yes.append(x41y)
credit_rating_yes.append(x42y)

x41n=df[(df['credit_rating']=='fair') & (df['Buy_Computer']=='no')]['credit_rating'].count()
x42n=df[(df['credit_rating']=='excellent') &
(df['Buy_Computer']=='no')]['credit_rating'].count()

credit_rating_no.append(x41n)
credit_rating_no.append(x42n)

print(credit_rating_yes,credit_rating_no)

for i in range(len(x4)):
  if (credit_rating_yes[i]>0):
     pp_credit_rating_yes.append(credit_rating_yes[i]/x4[i])
  else:
     pp_credit_rating_yes.append(0)
for i in range(len(x4)):
  if (credit_rating_no[i]>0):
    pp_credit_rating_no.append(credit_rating_no[i]/x3[i])
  else:
    pp_credit_rating_no.append(0)

print(pp_credit_rating_yes)
print(pp_credit_rating_no)

yes_evi=0

for i in pp_age_yes:
  yes_evi=yes_evi+(i*pp_target_yes)
for i in pp_income_yes:
  yes_evi=yes_evi+(i*pp_target_yes)
for i in pp_stu_yes:
  yes_evi=yes_evi+(i*pp_target_yes)
for i in pp_credit_rating_yes:
  yes_evi=yes_evi+(i*pp_target_yes)
print(yes_evi)

no_evi=0

for i in pp_age_no:
  no_evi=no_evi+(i*pp_target_no)
for i in pp_income_yes:
  no_evi=no_evi+(i*pp_target_no)
for i in pp_stu_no:
```

```
  no_evi=no_evi+(i*pp_target_no)
for i in pp_credit_rating_no:
  no_evi=no_evi+(i*pp_target_no)
print(no_evi)

evidence=yes_evi+no_evi
print(evidence)

#age='youth'
#income='medium'
#student='no'
#credit_rating='fair'

age='senior'
income='medium'
student='yes'
credit_rating='fair'
df1=[age,income,student,credit_rating]
age_order=x1.index.values
income_order=x2.index.values
stu_order=x3.index.values
cr_order=x4.index.values

import numpy as np
l1=np.where(age_order==age)[0]
l2=np.where(income_order==income)[0]
l3=np.where(stu_order==student)[0]
l4=np.where(cr_order==credit_rating)[0]
#print(l1,l2,l3,l4)
yes_val_nr=pp_age_yes[l1[0]]*pp_income_yes[l2[0]]*pp_stu_yes[l3[0]]*pp_credit_rating_y
es[l4[0]]
no_val_nr=pp_age_no[l1[0]]*pp_income_no[l2[0]]*pp_stu_no[l3[0]]*pp_credit_rating_no[l
4[0]]
yes_val=yes_val_nr/evidence
no_val=no_val_nr/evidence
if (yes_val>no_val):
  print(df1," - Buy_Computer status ==> ","yes")
else:
  print(df1," - Buy_Computer status ==> ","no")
```

INPUT / OUTPUT:



SET 1 EXP 4.ipynb

File Edit View Insert Runtime Tools Help

+ Code + Text

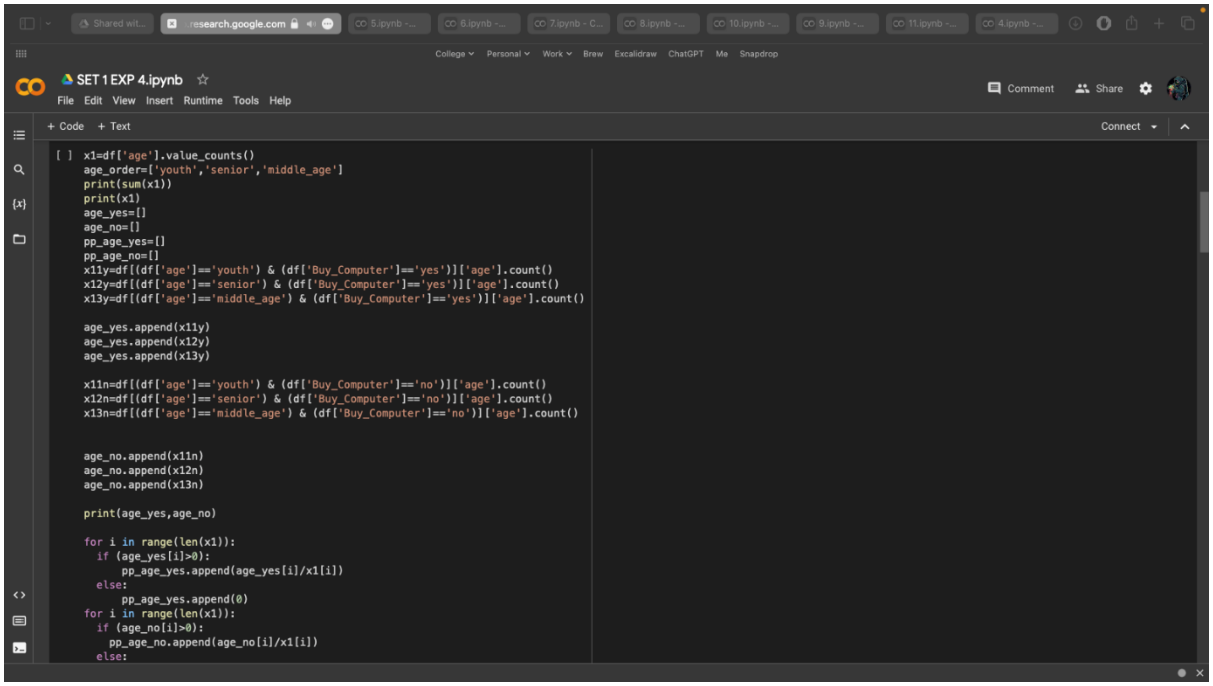
```
import pandas as pd
df = pd.read_csv('Buy_Computer.csv')
print(df)
```

```
    id         age  income student credit_rating Buy_Computer
0    1       youth    high      no          fair           no
1    2       youth    high      no     excellent           no
2    3  middle_age    high      no          fair          yes
3    4      senior  medium      no          fair          yes
4    5      senior     low     yes          fair          yes
5    6      senior     low     yes     excellent           no
6    7  middle_age     low     yes     excellent          yes
7    8       youth  medium      no          fair           no
8    9       youth     low     yes          fair          yes
9   10      senior  medium     yes          fair          yes
10  11       youth  medium     yes     excellent          yes
11  12  middle_age  medium      no     excellent          yes
12  13  middle_age    high     yes          fair          yes
13  14      senior  medium      no     excellent           no
```

```
x0y=df['Buy_Computer'].value_counts()['yes']
x0n=df['Buy_Computer'].value_counts()['no']
pp_target_yes=x0y/(x0y+x0n)
pp_target_no=x0n/(x0y+x0n)
print(x0y,x0n)
print(pp_target_yes,pp_target_no)
```

```
9 5
0.6428571428571429 0.3571428571428571
```

```
x1=df['age'].value_counts()
age_order=['youth','senior','middle_age']
print(sum(x1))
print(x1)
age_yes=[]
```

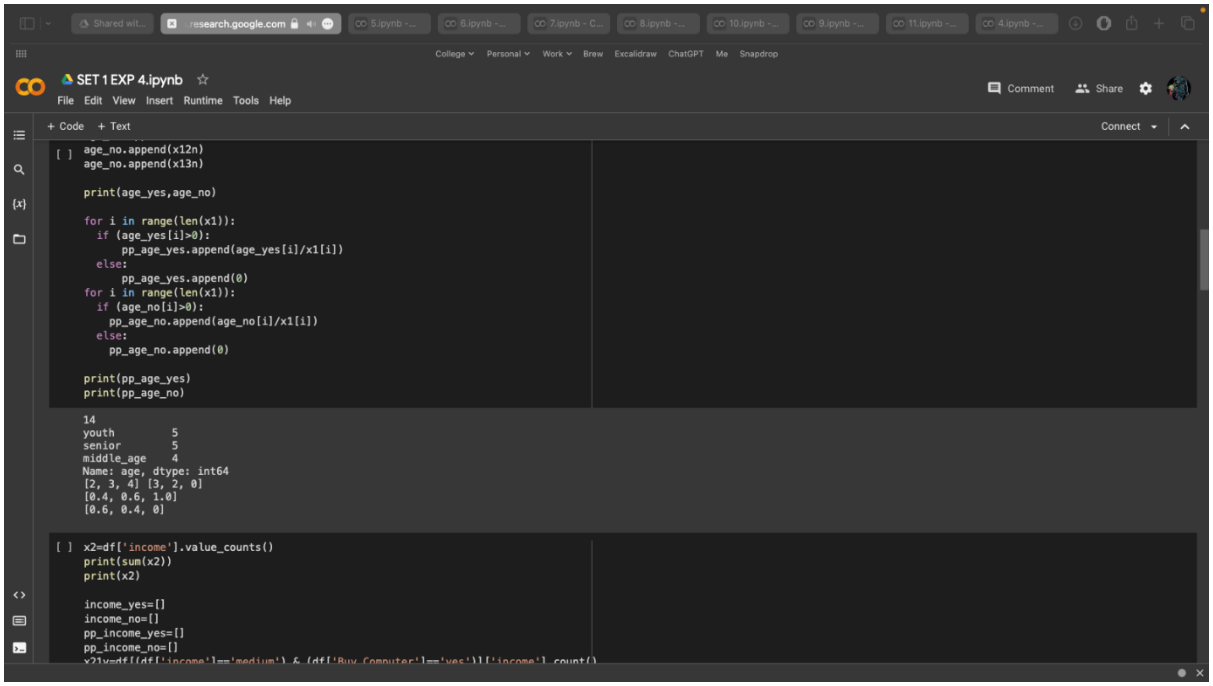


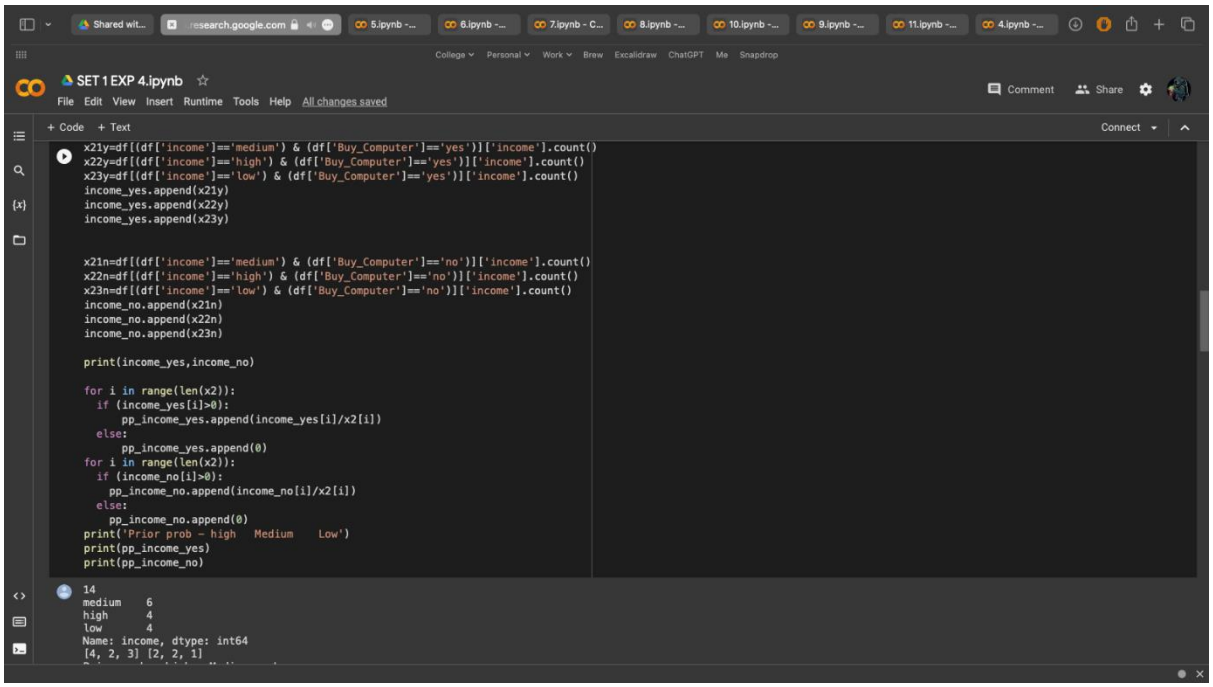
SET 1 EXP 4.ipynb

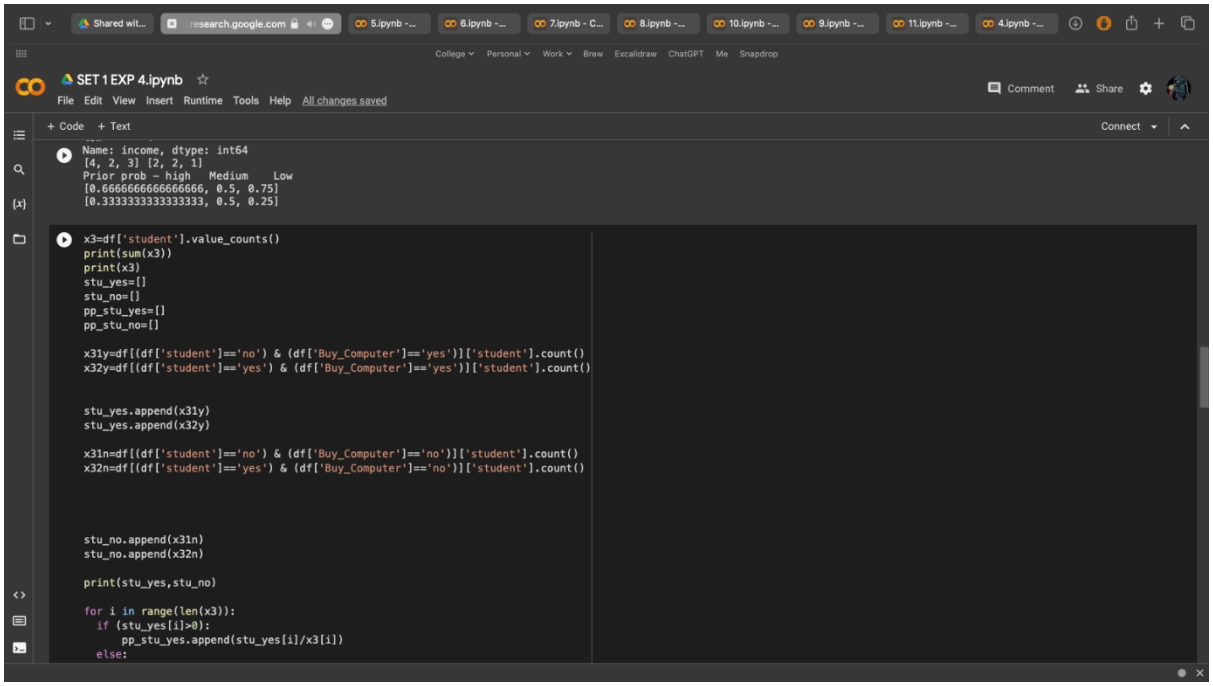
File Edit View Insert Runtime Tools Help

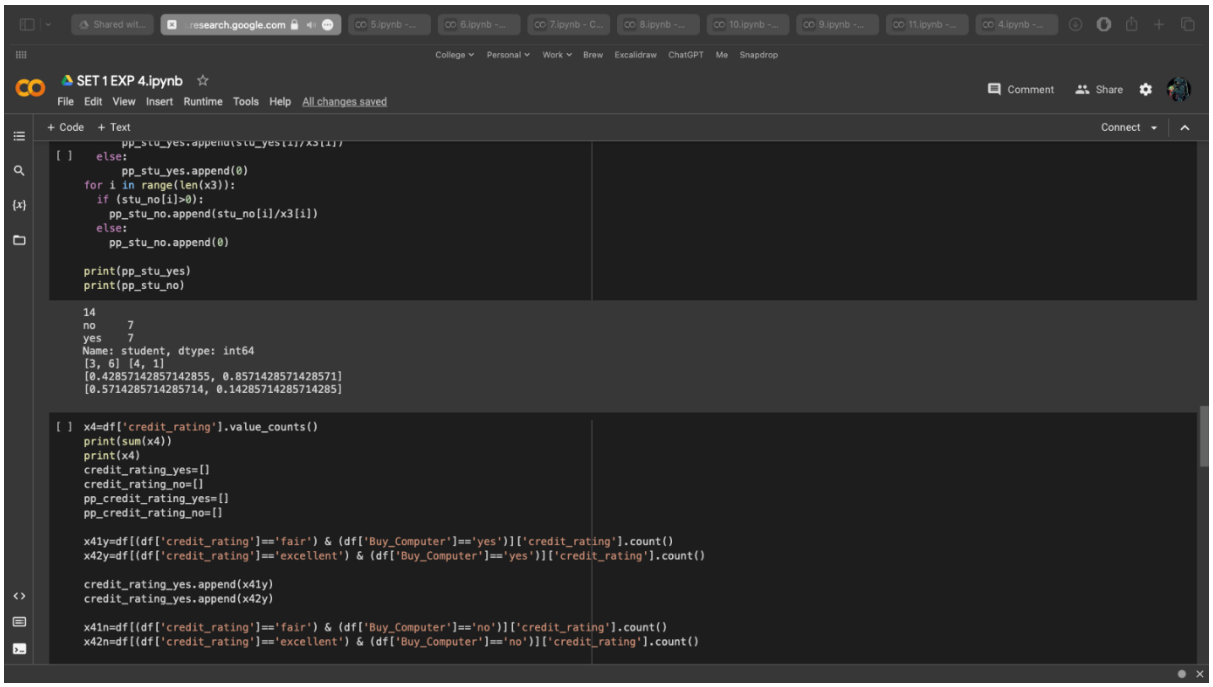
+ Code + Text

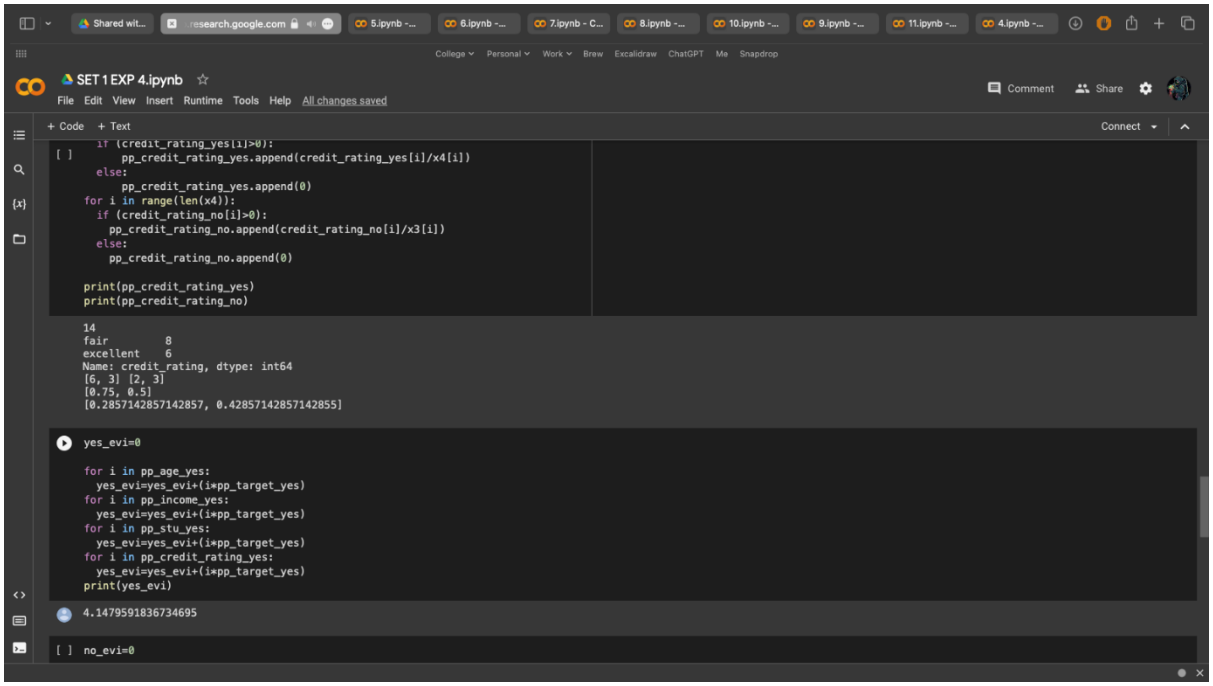
```
x1=df['age'].value_counts()
age_order=['youth','senior','middle_age']
print(sum(x1))
print(x1)
age_yes=[]
age_no=[]
pp_age_yes=[]
pp_age_no=[]
x11y=df[(df['age']=='youth') & (df['Buy_Computer']=='yes')]['age'].count()
x12y=df[(df['age']=='senior') & (df['Buy_Computer']=='yes')]['age'].count()
x13y=df[(df['age']=='middle_age') & (df['Buy_Computer']=='yes')]['age'].count()

age_yes.append(x11y)
age_yes.append(x12y)
age_yes.append(x13y)

x11n=df[(df['age']=='youth') & (df['Buy_Computer']=='no')]['age'].count()
x12n=df[(df['age']=='senior') & (df['Buy_Computer']=='no')]['age'].count()
x13n=df[(df['age']=='middle_age') & (df['Buy_Computer']=='no')]['age'].count()


age_no.append(x11n)
age_no.append(x12n)
age_no.append(x13n)

print(age_yes,age_no)

for i in range(len(x1)):
  if (age_yes[i]>0):
      pp_age_yes.append(age_yes[i]/x1[i])
  else:
      pp_age_yes.append(0)
for i in range(len(x1)):
  if (age_no[i]>0):
    pp_age_no.append(age_no[i]/x1[i])
  else:
```


SET 1 EXP 4.ipynb
File Edit View Insert Runtime Tools Help
Comment
Share
Connect
+ Code + Text
age_no.append(x12n)
age_no.append(x13n)
print(age_yes,age_no)
for i in range(len(x1)):
if (age_yes[i]>0):
pp_age_yes.append(age_yes[i]/x1[i])
else:
pp_age_yes.append(0)
for i in range(len(x1)):
if (age_no[i]>0):
pp_age_no.append(age_no[i]/x1[i])
else:
pp_age_no.append(0)
print(pp_age_yes)
print(pp_age_no)
14
youth 5
senior 5
middle_age 4
Name: age, dtype: int64
[2, 3, 4] [3, 2, 0]
[0.4, 0.6, 1.0]
[0.6, 0.4, 0]
x2=df['income'].value_counts()
print(sum(x2))
print(x2)
income_yes=[]
income_no=[]
pp_income_yes=[]
pp_income_no=[]


SET 1 EXP 4.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Connect
+ Code + Text
x21y=df[(df['income']=='medium') & (df['Buy_Computer']=='yes')]['income'].count()
x22y=df[(df['income']=='high') & (df['Buy_Computer']=='yes')]['income'].count()
x23y=df[(df['income']=='low') & (df['Buy_Computer']=='yes')]['income'].count()
income_yes.append(x21y)
income_yes.append(x22y)
income_yes.append(x23y)
x21n=df[(df['income']=='medium') & (df['Buy_Computer']=='no')]['income'].count()
x22n=df[(df['income']=='high') & (df['Buy_Computer']=='no')]['income'].count()
x23n=df[(df['income']=='low') & (df['Buy_Computer']=='no')]['income'].count()
income_no.append(x21n)
income_no.append(x22n)
income_no.append(x23n)
print(income_yes,income_no)
for i in range(len(x2)):
if (income_yes[i]>0):
pp_income_yes.append(income_yes[i]/x2[i])
else:
pp_income_yes.append(0)
for i in range(len(x2)):
if (income_no[i]>0):
pp_income_no.append(income_no[i]/x2[i])
else:
pp_income_no.append(0)
print('Prior prob - high Medium Low')
print(pp_income_yes)
print(pp_income_no)
14
medium 6
high 4
low 4
Name: income, dtype: int64
[4, 2, 3] [2, 2, 1]


SET 1 EXP 4.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Connect
+ Code + Text
Name: income, dtype: int64
[4, 2, 3] [2, 2, 1]
Prior prob - high Medium Low
[0.6666666666666666, 0.5, 0.75]
[0.3333333333333333, 0.5, 0.25]
x3=df['student'].value_counts()
print(sum(x3))
print(x3)
stu_yes=[]
stu_no=[]
pp_stu_yes=[]
pp_stu_no=[]
x31y=df[(df['student']=='no') & (df['Buy_Computer']=='yes')]['student'].count()
x32y=df[(df['student']=='yes') & (df['Buy_Computer']=='yes')]['student'].count()
stu_yes.append(x31y)
stu_yes.append(x32y)
x31n=df[(df['student']=='no') & (df['Buy_Computer']=='no')]['student'].count()
x32n=df[(df['student']=='yes') & (df['Buy_Computer']=='no')]['student'].count()
stu_no.append(x31n)
stu_no.append(x32n)
print(stu_yes,stu_no)
for i in range(len(x3)):
if (stu_yes[i]>0):
pp_stu_yes.append(stu_yes[i]/x3[i])
else:


SET 1 EXP 4.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Connect
+ Code + Text
else:
pp_stu_yes.append(0)
for i in range(len(x3)):
if (stu_no[i]>0):
pp_stu_no.append(stu_no[i]/x3[i])
else:
pp_stu_no.append(0)
print(pp_stu_yes)
print(pp_stu_no)
14
no 7
yes 7
Name: student, dtype: int64
[3, 6] [4, 1]
[0.42857142857142855, 0.8571428571428571]
[0.5714285714285714, 0.14285714285714285]
x4=df['credit_rating'].value_counts()
print(sum(x4))
print(x4)
credit_rating_yes=[]
credit_rating_no=[]
pp_credit_rating_yes=[]
pp_credit_rating_no=[]
x41y=df[(df['credit_rating']=='fair') & (df['Buy_Computer']=='yes')]['credit_rating'].count()
x42y=df[(df['credit_rating']=='excellent') & (df['Buy_Computer']=='yes')]['credit_rating'].count()
credit_rating_yes.append(x41y)
credit_rating_yes.append(x42y)
x41n=df[(df['credit_rating']=='fair') & (df['Buy_Computer']=='no')]['credit_rating'].count()
x42n=df[(df['credit_rating']=='excellent') & (df['Buy_Computer']=='no')]['credit_rating'].count()



SET 1 EXP 4.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text Connect

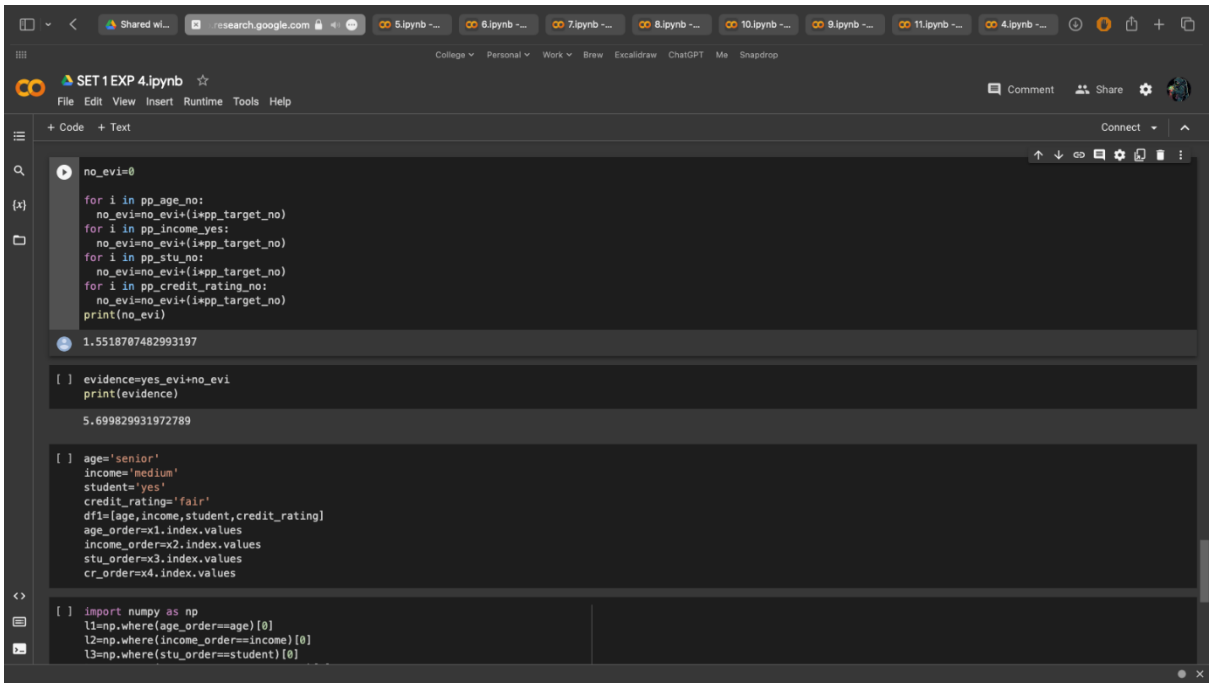
```
[ ]     if (credit_rating_yes[i]>0):
            pp_credit_rating_yes.append(credit_rating_yes[i]/x4[i])
        else:
            pp_credit_rating_yes.append(0)
    for i in range(len(x4)):
      if (credit_rating_no[i]>0):
        pp_credit_rating_no.append(credit_rating_no[i]/x3[i])
      else:
        pp_credit_rating_no.append(0)

    print(pp_credit_rating_yes)
    print(pp_credit_rating_no)
```

```
14
fair         8
excellent    6
Name: credit_rating, dtype: int64
[6, 3] [2, 3]
[0.75, 0.5]
[0.2857142857142857, 0.42857142857142855]
```

```
yes_evi=0

for i in pp_age_yes:
  yes_evi=yes_evi+(i*pp_target_yes)
for i in pp_income_yes:
  yes_evi=yes_evi+(i*pp_target_yes)
for i in pp_stu_yes:
  yes_evi=yes_evi+(i*pp_target_yes)
for i in pp_credit_rating_yes:
  yes_evi=yes_evi+(i*pp_target_yes)
print(yes_evi)
```

```
4.1479591836734695
```

```
[ ] no_evi=0
```



SET 1 EXP 4.ipynb

File Edit View Insert Runtime Tools Help

Comment Share

+ Code + Text Connect

```
no_evi=0

for i in pp_age_no:
  no_evi=no_evi+(i*pp_target_no)
for i in pp_income_yes:
  no_evi=no_evi+(i*pp_target_no)
for i in pp_stu_no:
  no_evi=no_evi+(i*pp_target_no)
for i in pp_credit_rating_no:
  no_evi=no_evi+(i*pp_target_no)
print(no_evi)
```

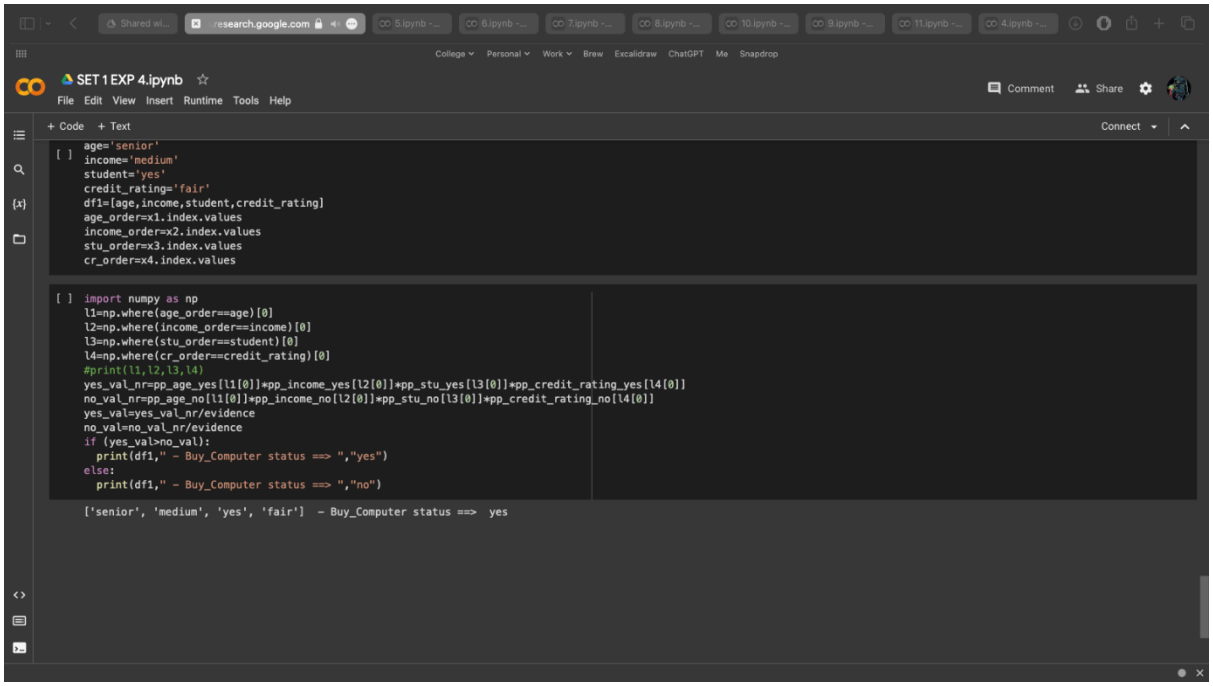
```
1.5518707482993197
```

```
[ ] evidence=yes_evi+no_evi
    print(evidence)
```

```
5.699829931972789
```

```
[ ] age='senior'
    income='medium'
    student='yes'
    credit_rating='fair'
    df1=[age,income,student,credit_rating]
    age_order=x1.index.values
    income_order=x2.index.values
    stu_order=x3.index.values
    cr_order=x4.index.values
```

```
[ ] import numpy as np
    l1=np.where(age_order==age)[0]
    l2=np.where(income_order==income)[0]
    l3=np.where(stu_order==student)[0]
```



RESULT

Thus, the Bayesian Network Classifier to perform machine learning classification in python has been implemented and executed successfully.

EXP. NO: 5 **DATE:**

ANN WITH BACK PROPAGATION

AIM

To implement the concepts of ANN Back Propagation in Machine Learning using Python.

ALGORITHM

1. Import necessary libraries: numpy, pandas, load_iris from sklearn.datasets, train_test_split from sklearn.model_selection, and matplotlib.pyplot.
2. Load the iris dataset using load_iris() method.
3. Get the features and target variables from the dataset by assigning X = data.data and y = data.target.
4. Get dummy variables for the target variable by using the pd.get_dummies(y).values method.
5. Split the data into training and testing datasets using train_test_split(X, y, test_size=20, random_state=4) method.
6. Set the learning rate to 0.1, number of iterations to 5000, number of input features to 4, number of hidden layer neurons to 2, and number of neurons at the output layer to 3.
7. Create an empty dataframe called results with columns "mse" and "accuracy".
8. Initialize the weights for the hidden layer and the output layer using np.random.normal(scale=0.5, size=(input_size, hidden_size)) and np.random.normal(scale=0.5, size=(hidden_size, output_size)) respectively.
9. Define the sigmoid function and the mean_squared_error function.
10. Run the iterations loop for the defined number of iterations. In each iteration perform necessary actions.

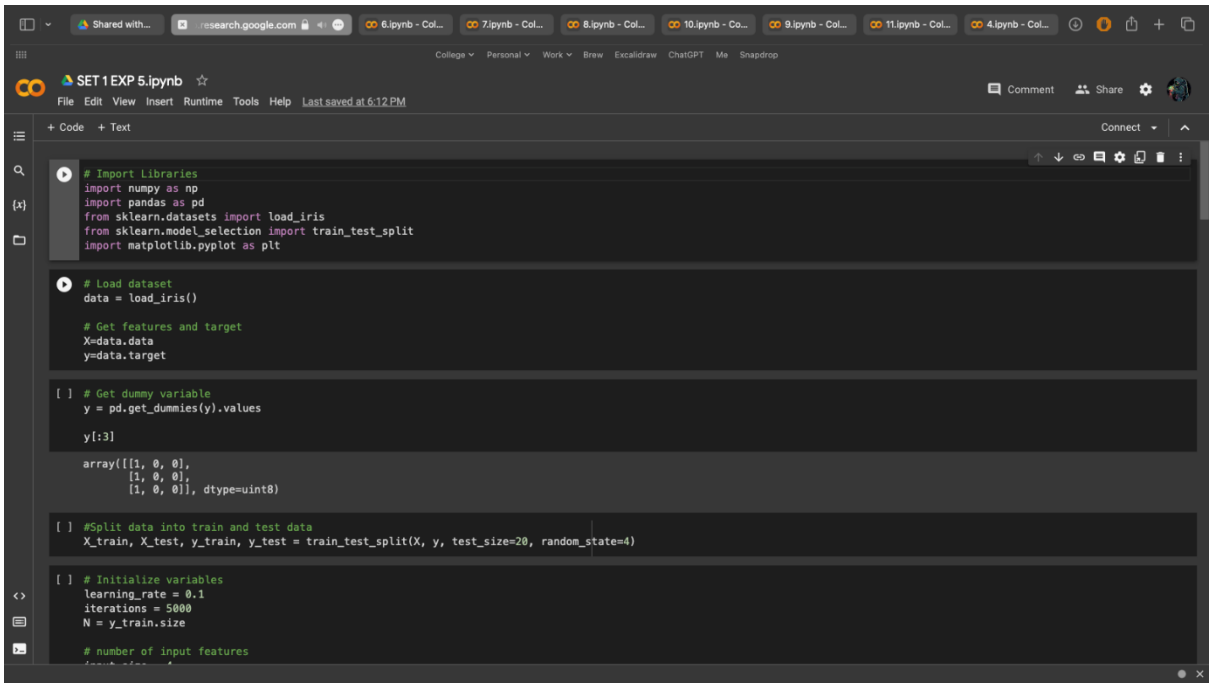
CODE SEGMENT

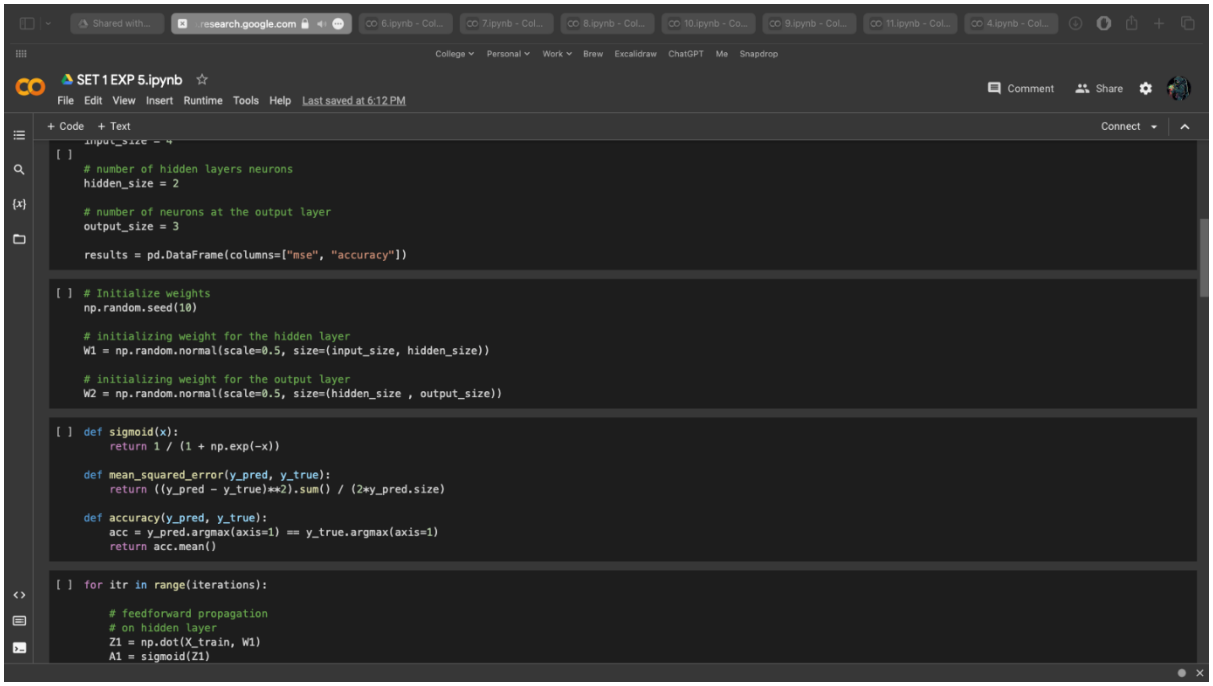
```
import numpy as np
import pandas as pd
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt

# Load dataset
data = load_iris()

# Get features and target
X=data.data
y=data.target

# Get dummy variable
y = pd.get_dummies(y).values
```

```
y[:3]

#Split data into train and test data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=20, random_state=4)

# Initialize variables
learning_rate = 0.1
iterations = 5000
N = y_train.size

# number of input features
input_size = 4

# number of hidden layers neurons
hidden_size = 2

# number of neurons at the output layer
output_size = 3

results = pd.DataFrame(columns=["mse", "accuracy"])

# Initialize weights
np.random.seed(10)

# initializing weight for the hidden layer
W1 = np.random.normal(scale=0.5, size=(input_size, hidden_size))

# initializing weight for the output layer
W2 = np.random.normal(scale=0.5, size=(hidden_size , output_size))

def sigmoid(x):
    return 1 / (1 + np.exp(-x))

def mean_squared_error(y_pred, y_true):
    return ((y_pred - y_true)**2).sum() / (2*y_pred.size)

def accuracy(y_pred, y_true):
    acc = y_pred.argmax(axis=1) == y_true.argmax(axis=1)
    return acc.mean()

for itr in range(iterations):

    # feedforward propagation
    # on hidden layer
    Z1 = np.dot(X_train, W1)
    A1 = sigmoid(Z1)
```

```
    # on output layer
    Z2 = np.dot(A1, W2)
    A2 = sigmoid(Z2)


    # Calculating error
    mse = mean_squared_error(A2, y_train)
    acc = accuracy(A2, y_train)
    results=results.append({"mse":mse, "accuracy":acc},ignore_index=True )

    # backpropagation
    E1 = A2 - y_train
    dW1 = E1 * A2 * (1 - A2)

    E2 = np.dot(dW1, W2.T)
    dW2 = E2 * A1 * (1 - A1)


    # weight updates
    W2_update = np.dot(A1.T, dW1) / N
    W1_update = np.dot(X_train.T, dW2) / N

    W2 = W2 - learning_rate * W2_update
    W1 = W1 - learning_rate * W1_update

results.mse.plot(title="Mean Squared Error")

results.accuracy.plot(title="Accuracy")

# feedforward
Z1 = np.dot(X_test, W1)
A1 = sigmoid(Z1)

Z2 = np.dot(A1, W2)
A2 = sigmoid(Z2)

acc = accuracy(A2, y_test)
print("Accuracy: {}".format(acc))
```

INPUT / OUTPUT





SET 1 EXP 5.ipynb

File Edit View Insert Runtime Tools Help Last saved at 6:12 PM

+ Code + Text

```
def accuracy(y_pred, y_true):
    acc = y_pred.argmax(axis=1) == y_true.argmax(axis=1)
    return acc.mean()
```

```
for itr in range(iterations):

    # feedforward propagation
    # on hidden layer
    Z1 = np.dot(X_train, W1)
    A1 = sigmoid(Z1)

    # on output layer
    Z2 = np.dot(A1, W2)
    A2 = sigmoid(Z2)


    # Calculating error
    mse = mean_squared_error(A2, y_train)
    acc = accuracy(A2, y_train)
    results=results.append({"mse":mse, "accuracy":acc},ignore_index=True )

    # backpropagation
    E1 = A2 - y_train
    dW1 = E1 * A2 * (1 - A2)

    E2 = np.dot(dW1, W2.T)
    dW2 = E2 * A1 * (1 - A1)


    # weight updates
    W2_update = np.dot(A1.T, dW1) / N
    W1_update = np.dot(X_train.T, dW2) / N

    W2 = W2 - learning_rate * W2_update
    W1 = W1 - learning_rate * W1_update
```

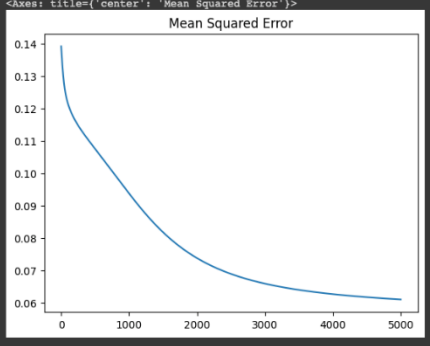
SET 1 EXP 5.ipynb

File Edit View Insert Runtime Tools Help Last saved at 6:12 PM

+ Code + Text

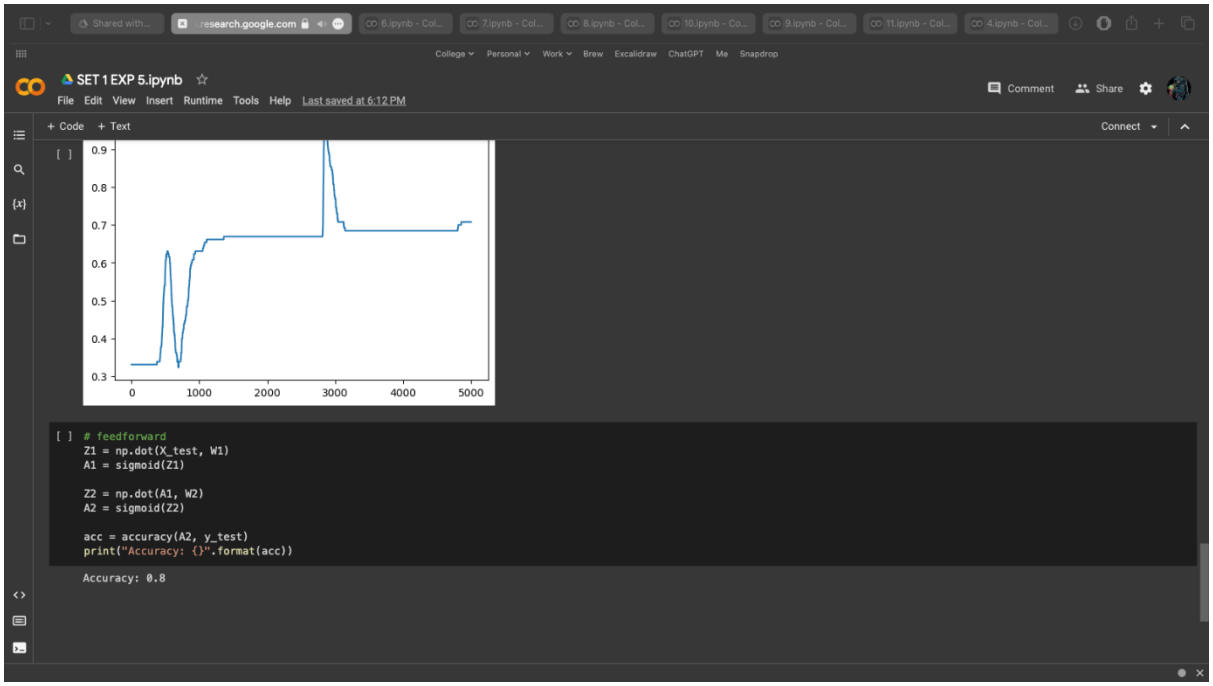
```
results.mse.plot(title="Mean Squared Error")
```

```
<Axes: title={'center': 'Mean Squared Error'}>
```




```
[ ] results.accuracy.plot(title="Accuracy")
```

```
<Axes: title={'center': 'Accuracy'}>
```



RESULT

Thus, the concept of ANN Back Propagation in Machine Learning using python has been implemented and executed successfully.

EXP. NO: 6 **DATE:**

CLASSIFICATION USING DECISION TREE

AIM

To perform machine learning - classification in python using Decision Tree Classifier.

ALGORITHM

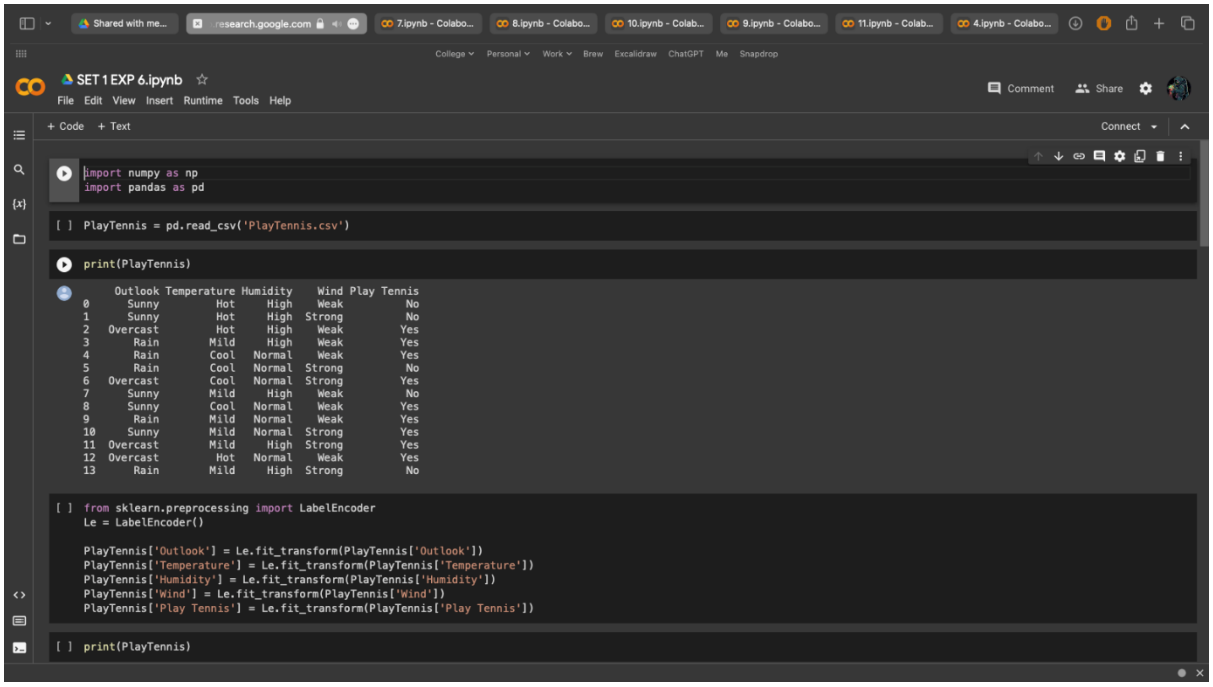
1. Import numpy and pandas libraries.
2. Import the necessary dataset using the pandas read_csv() function.
3. Import LabelEncoder from sklearn.preprocessing library.
4. Apply LabelEncoder to each categorical feature of the dataset using fit_transform() function of LabelEncoder.
5. Split the dataset into target variable y and input features X.
6. Import DecisionTreeClassifier from sklearn.tree library.
7. Initialize the DecisionTreeClassifier with criterion as entropy.
8. Fit the DecisionTreeClassifier model with the input features X and target variable y using the fit() function.
9. Visualize the decision tree using tree.plot_tree() function.
10. Predict the values of target variable using the DecisionTreeClassifier model and compare the predicted values with the actual target variable to check the accuracy of the model.

CODE SEGMENT

```
import numpy as np
import pandas as pd

from google.colab import drive

drive.mount('/content/drive', force_remount=True)

!cd 'drive/My Drive/Lab Programs/Ex 6/'

PlayTennis = pd.read_csv('drive/My Drive/Lab Programs/Ex 6/PlayTennis.csv')

from sklearn.preprocessing import LabelEncoder
Le = LabelEncoder()

PlayTennis['Outlook'] = Le.fit_transform(PlayTennis['Outlook'])
PlayTennis['Temperature'] = Le.fit_transform(PlayTennis['Temperature'])
PlayTennis['Humidity'] = Le.fit_transform(PlayTennis['Humidity'])
PlayTennis['Wind'] = Le.fit_transform(PlayTennis['Wind'])
PlayTennis['Play Tennis'] = Le.fit_transform(PlayTennis['Play Tennis'])
```

```
print(PlayTennis)

y = PlayTennis['Play Tennis']
X = PlayTennis.drop(['Play Tennis'],axis=1)

from sklearn import tree
clf = tree.DecisionTreeClassifier(criterion = 'entropy')
clf = clf.fit(X, y)

tree.plot_tree(clf)

X_pred = clf.predict(X)

# verifying if the model has predicted it all right.
X_pred == y
```

INPUT / OUTPUT



SET 1 EXP 6.ipynb

File Edit View Insert Runtime Tools Help

+ Code + Text

```
print(PlayTennis)
```

```
    Outlook  Temperature  Humidity  Wind  Play Tennis
0         2            1         0     1            0
1         2            1         0     0            0
2         0            1         0     1            1
3         1            2         0     1            1
4         1            0         1     1            1
5         1            0         1     0            0
6         0            0         1     0            1
7         2            2         0     1            0
8         2            0         1     1            1
9         1            2         1     1            1
10        2            2         1     0            1
11        0            2         0     0            1
12        0            1         1     1            1
13        1            2         0     0            0
```

```
y = PlayTennis['Play Tennis']
X = PlayTennis.drop(['Play Tennis'],axis=1)
```

```
from sklearn import tree
clf = tree.DecisionTreeClassifier(criterion = 'entropy')
clf = clf.fit(X, y)
```

```
tree.plot_tree(clf)
```

```
[Text(0.4444444444444444, 0.9, 'x[0] <= 0.5\nentropy = 0.94\nsamples = 14\nvalue = [5, 9]'),
 Text(0.3333333333333333, 0.7, 'entropy = 0.0\nsamples = 4\nvalue = [0, 4]'),
 Text(0.5555555555555556, 0.7, 'x[2] <= 0.5\nentropy = 1.0\nsamples = 10\nvalue = [5, 5]'),
 Text(0.3333333333333333, 0.5, 'x[0] <= 1.5\nentropy = 0.722\nsamples = 5\nvalue = [4, 1]'),
 Text(0.2222222222222222, 0.3, 'x[3] <= 0.5\nentropy = 1.0\nsamples = 2\nvalue = [1, 1]'),
 Text(0.1111111111111111, 0.1, 'entropy = 0.0\nsamples = 1\nvalue = [1, 0]'),
 Text(0.3333333333333333, 0.1, 'entropy = 0.0\nsamples = 1\nvalue = [0, 1]'),
 Text(0.4444444444444444, 0.3, 'entropy = 0.0\nsamples = 3\nvalue = [3, 0]'),
```

SET 1 EXP 6.ipynb

File Edit View Insert Runtime Tools Help Last saved at 6:26 PM

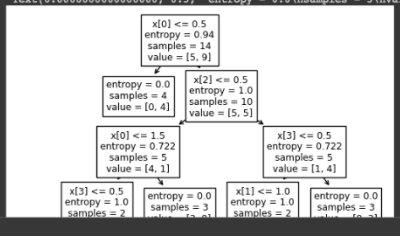
+ Code + Text

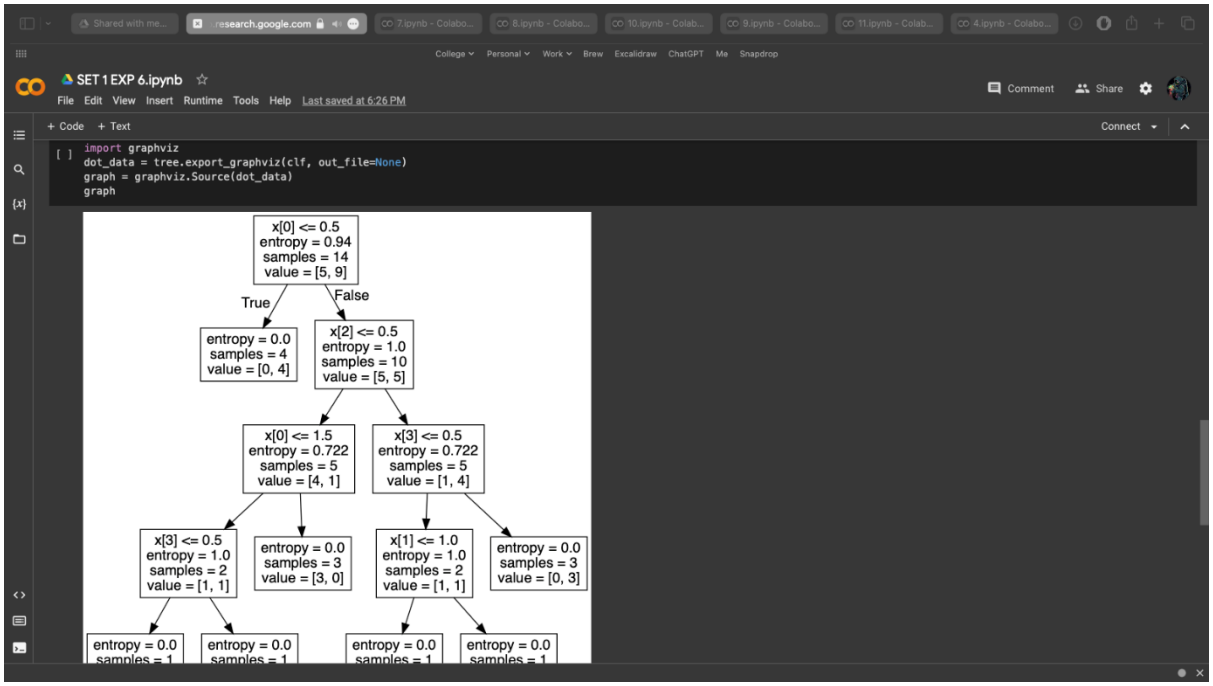
```
y = PlayTennis['Play Tennis']
X = PlayTennis.drop(['Play Tennis'],axis=1)
```

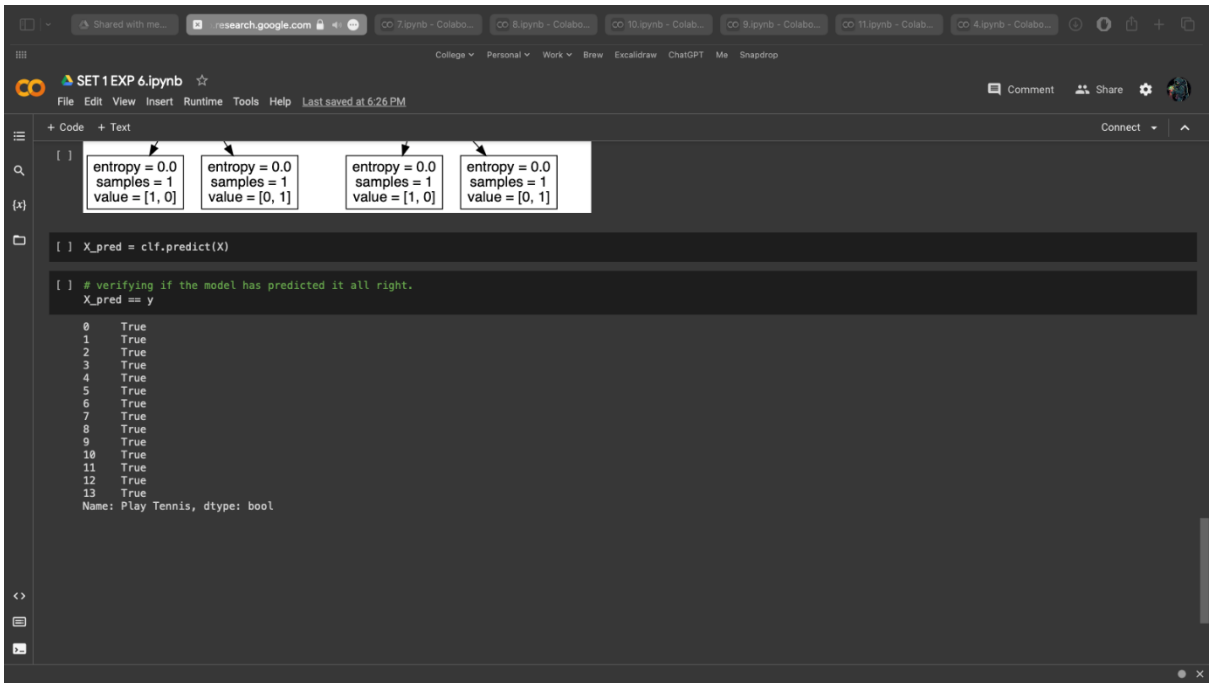
```
from sklearn import tree
clf = tree.DecisionTreeClassifier(criterion = 'entropy')
clf = clf.fit(X, y)
```

```
tree.plot_tree(clf)
```

```
[Text(0.4444444444444444, 0.9, 'x[0] <= 0.5\nentropy = 0.94\nsamples = 14\nvalue = [5, 9]'),
 Text(0.3333333333333333, 0.7, 'entropy = 0.0\nsamples = 4\nvalue = [0, 4]'),
 Text(0.5555555555555556, 0.7, 'x[2] <= 0.5\nentropy = 1.0\nsamples = 10\nvalue = [5, 5]'),
 Text(0.3333333333333333, 0.5, 'x[0] <= 1.5\nentropy = 0.722\nsamples = 5\nvalue = [4, 1]'),
 Text(0.2222222222222222, 0.3, 'x[3] <= 0.5\nentropy = 1.0\nsamples = 2\nvalue = [1, 1]'),
 Text(0.1111111111111111, 0.1, 'entropy = 0.0\nsamples = 1\nvalue = [1, 0]'),
 Text(0.3333333333333333, 0.1, 'entropy = 0.0\nsamples = 1\nvalue = [0, 1]'),
 Text(0.4444444444444444, 0.3, 'entropy = 0.0\nsamples = 3\nvalue = [3, 0]'),
 Text(0.7777777777777778, 0.5, 'x[3] <= 0.5\nentropy = 0.722\nsamples = 5\nvalue = [1, 4]'),
 Text(0.6666666666666666, 0.3, 'x[1] <= 1.0\nentropy = 1.0\nsamples = 2\nvalue = [1, 1]'),
 Text(0.5555555555555556, 0.1, 'entropy = 0.0\nsamples = 1\nvalue = [1, 0]'),
 Text(0.7777777777777778, 0.1, 'entropy = 0.0\nsamples = 1\nvalue = [0, 1]'),
 Text(0.8888888888888888, 0.3, 'entropy = 0.0\nsamples = 3\nvalue = [0, 3]')]
```







RESULT

Thus, the machine learning - classification in python using the Decision tree classifier has been implemented and executed successfully.

EXP. NO: 7 **DATE:**

REGRESSION USING DECISION TREE

AIM

To perform machine learning - regression in python using Decision Tree Classifier.

ALGORITHM

1. Import pandas library as pd.
2. Load the csv file named 'Train.csv' into a pandas dataframe 'df' using pd.read_csv('Train.csv').
3. Print the first five rows of the dataframe using df.head().
4. Check for missing values in the dataframe using df.isnull().sum(axis=0).
5. Count the frequency of each unique value in the column 'Item_Fat_Content' using df.Item_Fat_Content.value_counts().
6. Count the frequency of each unique value in the column 'Item_Visibility' and print the first five rows using df.Item_Visibility.value_counts().head().
7. Count the frequency of each unique value in the column 'Outlet_Size' using df.Outlet_Size.value_counts().
8. Create a cross-tabulation table between 'Outlet_Size' and 'Outlet_Type' columns using pd.crosstab(df['Outlet_Size'], df['Outlet_Type']) and store it in 'crosstable'.
9. Create a dictionary named 'dic' with the key-value pair {'Grocery Store':'Small'}.
10. Map the dictionary 'dic' to the 'Outlet_Type' column of the dataframe using s = df.Outlet_Type.map(dic).
11. Combine the 'Outlet_Size' column and 's' column using df.Outlet_Size.combine_first(s).
12. Count the frequency of each unique value in the 'Outlet_Size' column of the dataframe using df.Outlet_Size.value_counts().
13. Create a dictionary named 'dic' with the key-value pair {"Tier 2":"Small"}.
14. Map the dictionary 'dic' to the 'Outlet_Location_Type' column of the dataframe using s = df.Outlet_Location_Type.map(dic).
15. Replace the missing values in the 'Item_Weight' column by the mean of the column grouped by 'Item_Identifier' using df['Item_Weight'] = df['Item_Weight'].fillna(df.groupby('Item_Identifier')['Item_Weight'].transform('mean')).

CODE SEGMENT

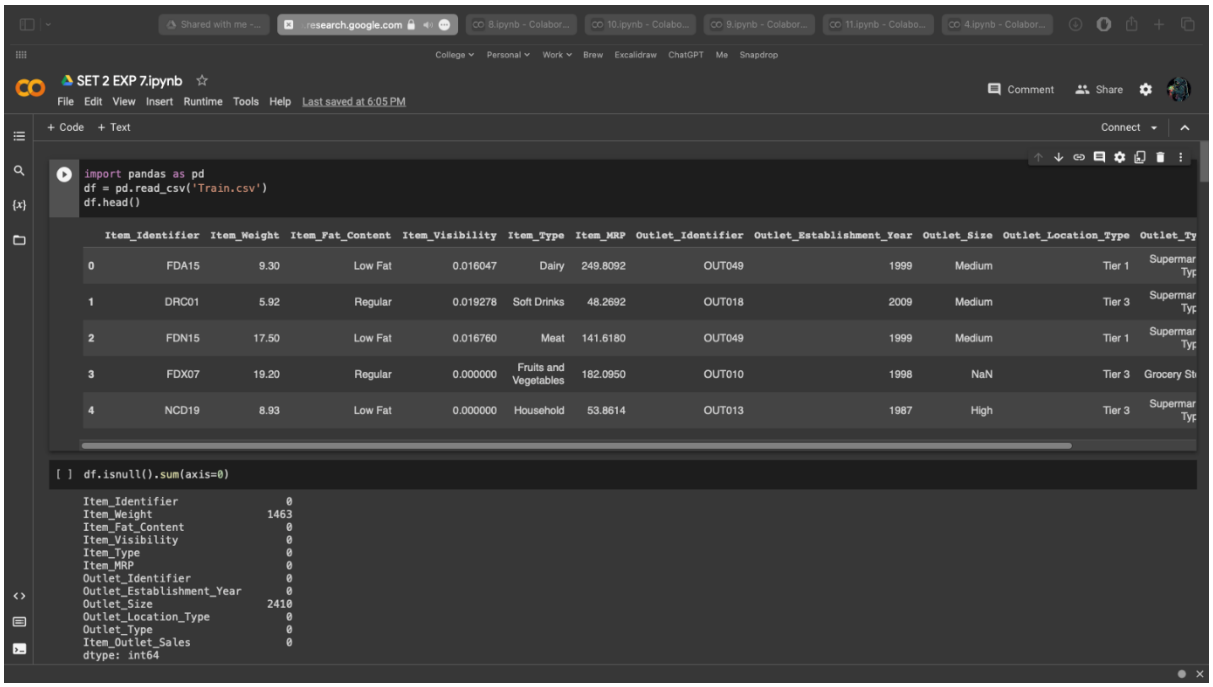
```
import pandas as pd
df = pd.read_csv('Train.csv')
df.head()

df.isnull().sum(axis=0)

df.Item_Fat_Content.value_counts() # has mismatched factor levels
```

```
df.Item_Visibility.value_counts().head()

df.Outlet_Size.value_counts()

crosstable = pd.crosstab(df['Outlet_Size'],df['Outlet_Type'])
dic = {'Grocery Store':'Small'}
s = df.Outlet_Type.map(dic)
df.Outlet_Size= df.Outlet_Size.combine_first(s)
df.Outlet_Size.value_counts()

# Checking if imputation was successful
df.isnull().sum(axis=0)

dic = {"Tier 2":"Small"}
s = df.Outlet_Location_Type.map(dic)
df.Outlet_Size = df.Outlet_Size.combine_first(s)

df['Item_Weight']=df['Item_Weight'].fillna(df.groupby('Item_Identifier')['Item_Weight'].trans
form('mean'))
df.isnull().sum()

df[df.Item_Weight.isnull()]

# List of item types
item_type_list = df.Item_Type.unique().tolist()
# grouping based on item type and calculating mean of item weight
Item_Type_Means = df.groupby('Item_Type')['Item_Weight'].mean()
# Mapiing Item weight to item type mean
for i in item_type_list:
  dic = {i:Item_Type_Means[i]}
  s = df.Item_Type.map(dic)
  df.Item_Weight = df.Item_Weight.combine_first(s)

Item_Type_Means = df.groupby('Item_Type')['Item_Weight'].mean()
# Checking if Imputation was successful
df.isnull().sum()

# Replacing 0's with NaN
import numpy as np
df.Item_Visibility.replace(to_replace=0.000000,value=np.NaN,inplace=True)
# Now fill by mean of visbility based on item identifiers
df.Item_Visibility =
df.Item_Visibility.fillna(df.groupby('Item_Identifier')['Item_Visibility'].transform('mean'))
# Checking if Imputation was carried out successfully
df.isnull().sum()
```

```
df.Item_Fat_Content.value_counts()

df.Item_Fat_Content.replace(to_replace=["LF","low fat"],value="Low Fat",inplace=True)
df.Item_Fat_Content.replace(to_replace="reg",value="Regular",inplace=True)
df.Item_Fat_Content.value_counts()

df['Outlet_Year'] = (2013 - df.Outlet_Establishment_Year)
df.head()

var_cat = df.select_dtypes(include=[object])
var_cat.head()

#Convert categorical into numerical
var_cat = var_cat.columns.tolist()
var_cat = ['Item_Fat_Content',
'Item_Type',
'Outlet_Size',
'Outlet_Location_Type',
'Outlet_Type']
var_cat
['Item_Fat_Content',
'Item_Type',
'Outlet_Size',
'Outlet_Location_Type',
'Outlet_Type']

import re
df.Item_Type.replace(to_replace="^FD*.*",value="Food",regex=True,inplace=True)
df.Item_Type.replace(to_replace="^DR*.*",value="Drinks",regex=True,inplace=True)
df.Item_Type.replace(to_replace="^NC*.*",value="Non-
Consumable",regex=True,inplace=True)
df.head()

from sklearn import preprocessing
le = preprocessing.LabelEncoder()
df['Outlet'] = le.fit_transform(df.Outlet_Identifier)
df['Item'] = le.fit_transform(df.Item_Type)
df.head()

for i in var_cat:
  df[i] = le.fit_transform(df[i])
df.head()

predictors=['Item_Fat_Content','Item_Visibility','Item_Type','Item_MRP','Outlet_Size','Outlet
_Location_Type','Outlet_Type','Outlet_Year',
```

```
'Outlet','Item','Item_Weight']
seed = 240
np.random.seed(seed)
X = df[predictors]
y = df.Item_Outlet_Sales
X.head()

from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size = 0.25,random_state = 42)
X_train.shape

X_train.tail()

from sklearn.tree import DecisionTreeRegressor
import sklearn.metrics as metrics
regressor = DecisionTreeRegressor(random_state=0)
regressor.fit(X_train,y_train)
DecisionTreeRegressor(criterion='mse', max_depth=None, max_features=None,
max_leaf_nodes=None, min_impurity_decrease=0.0,
min_samples_leaf=1,
min_samples_split=2, min_weight_fraction_leaf=0.0,
random_state=0, splitter='best')
predictions = regressor.predict(X_test)
print(predictions[:5])
rmse = np.sqrt(metrics.mean_squared_error(y_test,predictions))
print("RMSE = ",rmse)
results = pd.DataFrame({'Actual':y_test,'Predicted':predictions})
results.head()
```

INPUT / OUTPUT



SET 2 EXP 7.ipynb

File Edit View Insert Runtime Tools Help Last saved at 6:05 PM

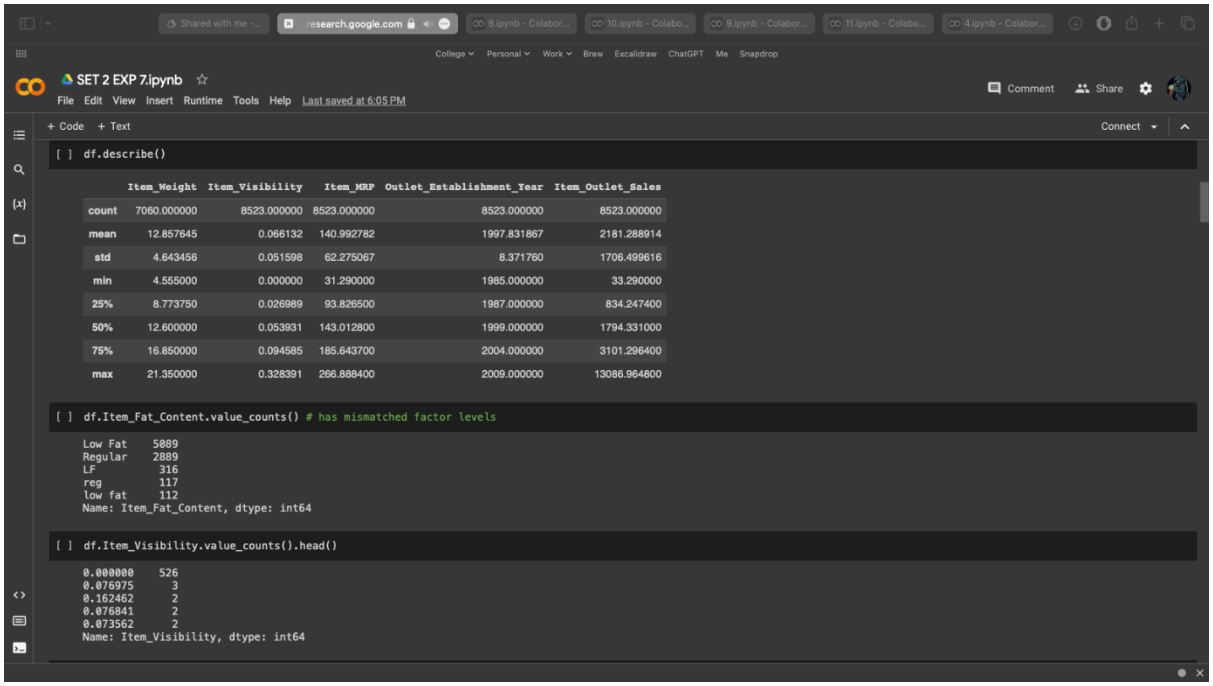
+ Code + Text

```
import pandas as pd
df = pd.read_csv('Train.csv')
df.head()
```

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Outlet_Size	Outlet_Location_Type	Outlet_Ty
0	FDA15	9.30	Low Fat	0.016047	Dairy	249.8092	OUT049	1999	Medium	Tier 1	Supermar Typ
1	DRC01	5.92	Regular	0.019278	Soft Drinks	48.2692	OUT018	2009	Medium	Tier 3	Supermar Typ
2	FDN15	17.50	Low Fat	0.016760	Meat	141.6180	OUT049	1999	Medium	Tier 1	Supermar Typ
3	FDX07	19.20	Regular	0.000000	Fruits and Vegetables	182.0950	OUT010	1998	NaN	Tier 3	Grocery St
4	NCD19	8.93	Low Fat	0.000000	Household	53.8614	OUT013	1987	High	Tier 3	Supermar Typ

```
df.isnull().sum(axis=0)
```

```
Item_Identifier               0
Item_Weight                1463
Item_Fat_Content              0
Item_Visibility               0
Item_Type                     0
Item_MRP                      0
Outlet_Identifier             0
Outlet_Establishment_Year     0
Outlet_Size                2410
Outlet_Location_Type          0
Outlet_Type                   0
Item_Outlet_Sales             0
dtype: int64
```



SET 2 EXP 7.ipynb

File Edit View Insert Runtime Tools Help Last saved at 6:05 PM

+ Code + Text

```
df.describe()
```

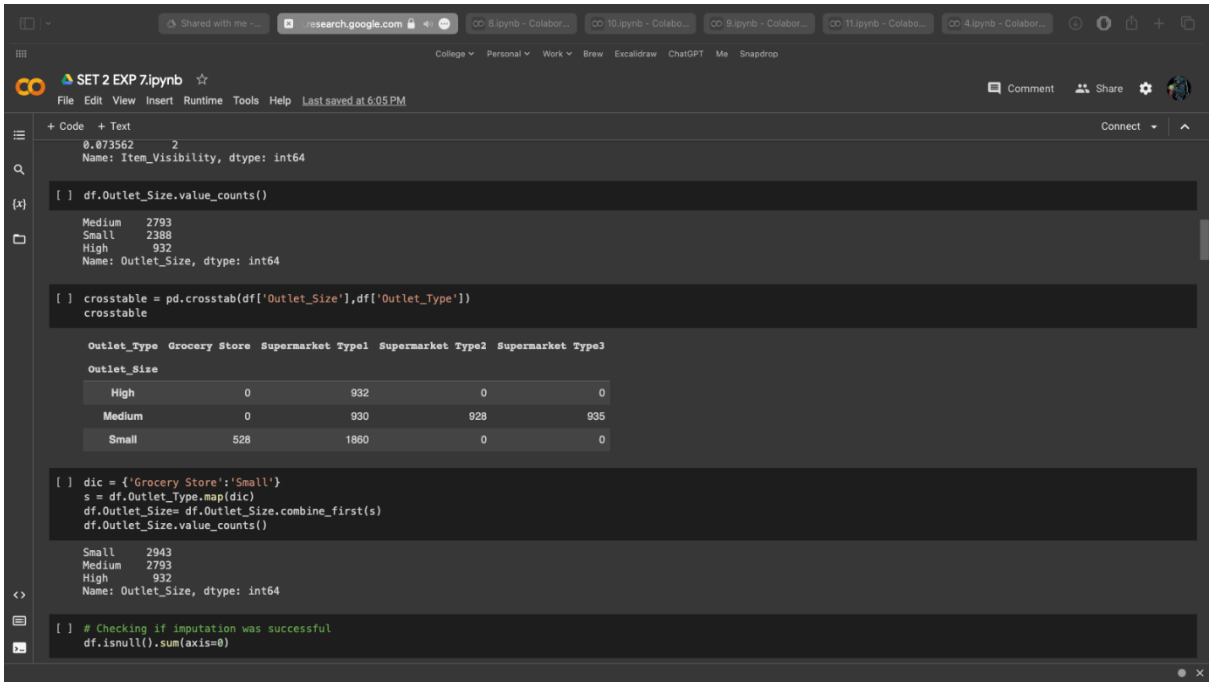
	Item_Weight	Item_Visibility	Item_MRP	Outlet_Establishment_Year	Item_Outlet_Sales
count	7060.000000	8523.000000	8523.000000	8523.000000	8523.000000
mean	12.857645	0.066132	140.992782	1997.831867	2181.288914
std	4.643456	0.051598	62.275067	8.371760	1706.499616
min	4.555000	0.000000	31.290000	1985.000000	33.290000
25%	8.773750	0.026989	93.826500	1987.000000	834.247400
50%	12.600000	0.053931	143.012800	1999.000000	1794.331000
75%	16.850000	0.094585	185.643700	2004.000000	3101.296400
max	21.350000	0.328391	266.888400	2009.000000	13086.964800

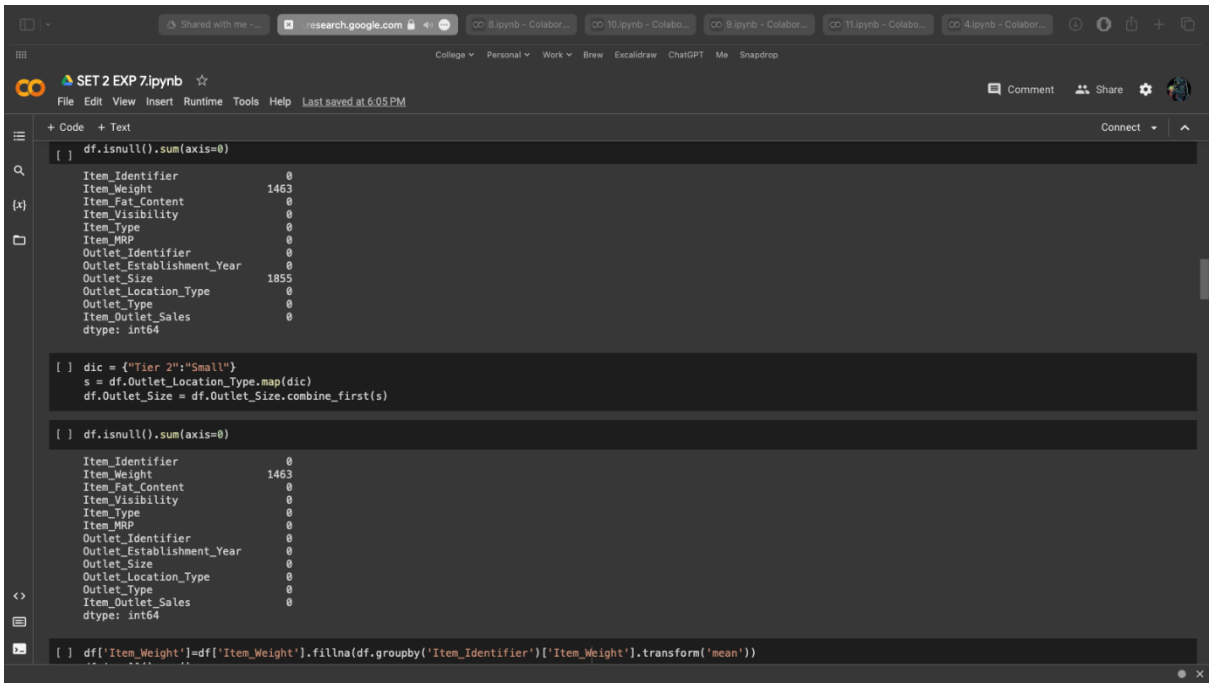
```
df.Item_Fat_Content.value_counts() # has mismatched factor levels
```

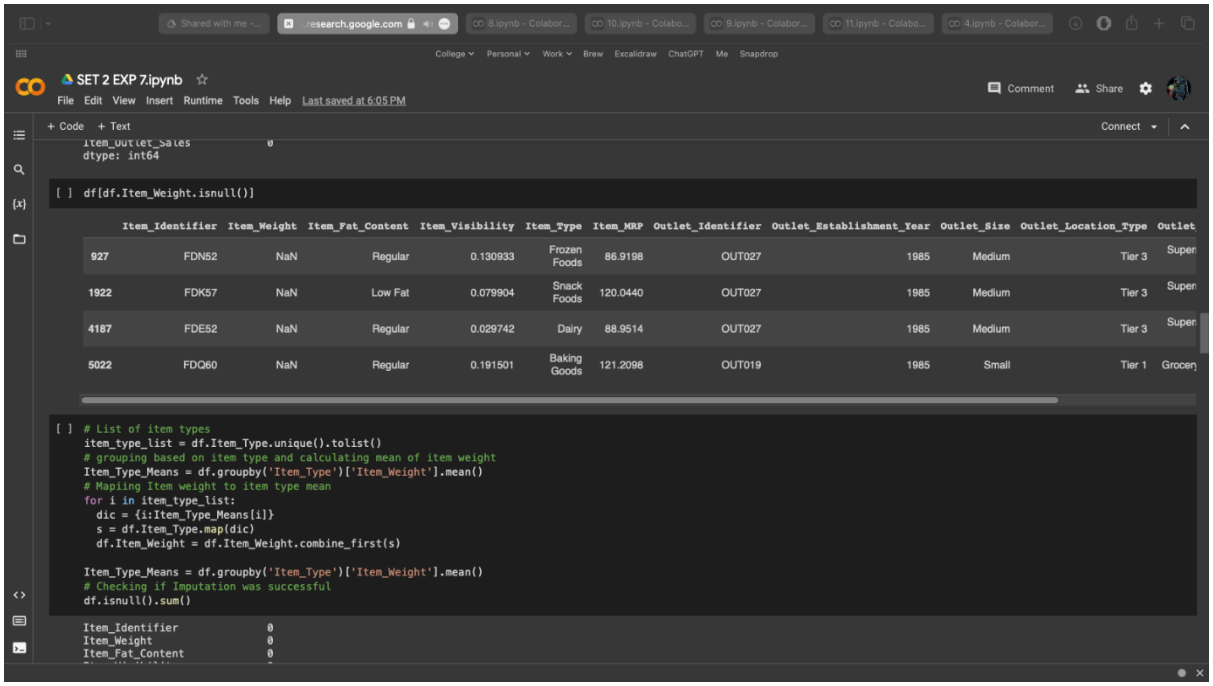
```
Low Fat    5089
Regular    2889
LF          316
reg         117
low fat     112
Name: Item_Fat_Content, dtype: int64
```

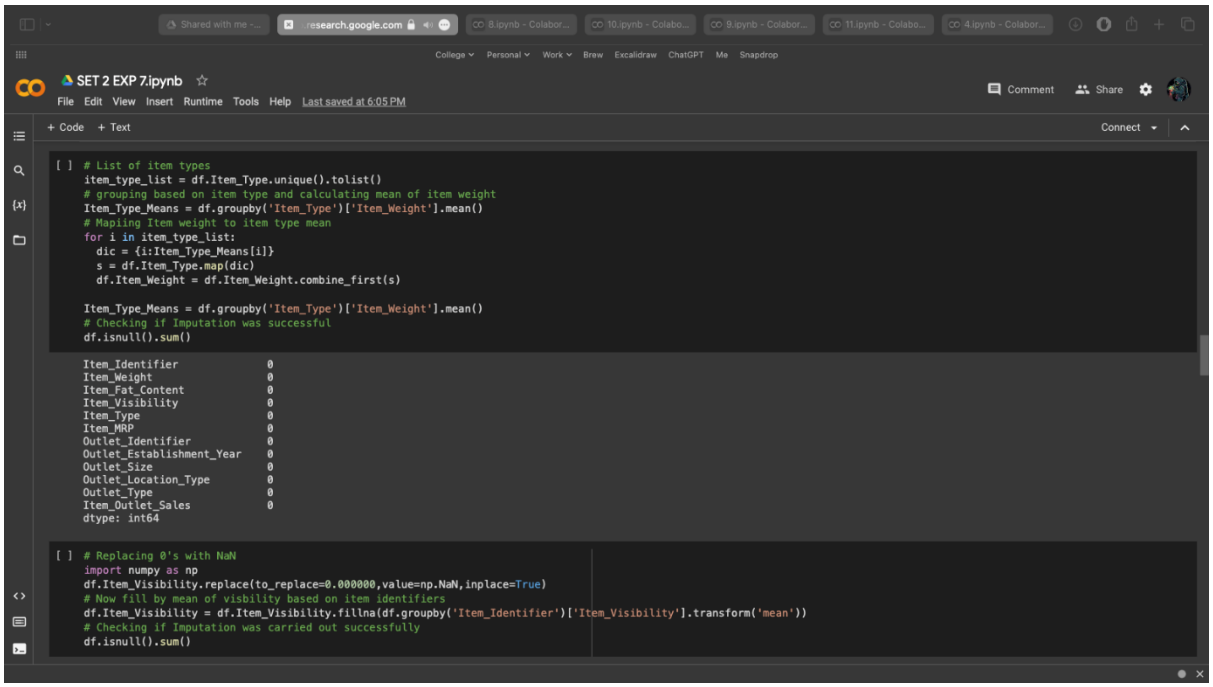
```
df.Item_Visibility.value_counts().head()
```

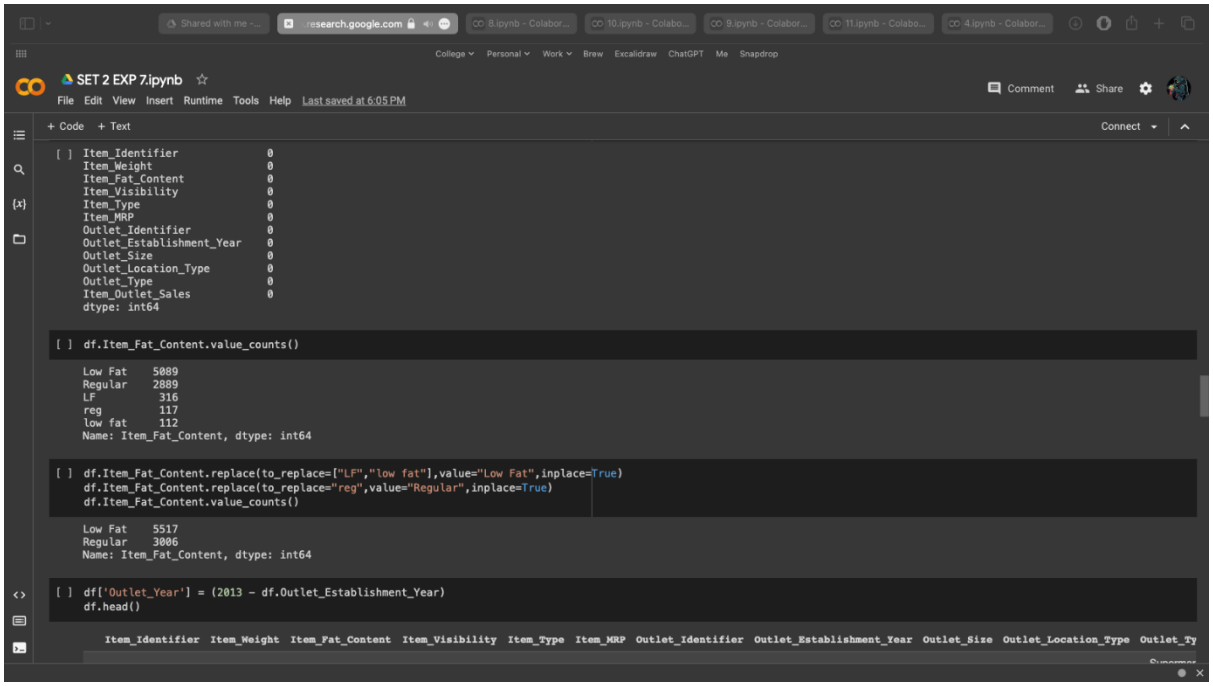
```
0.000000    526
0.076975      3
0.162462      2
0.076841      2
0.073562      2
Name: Item_Visibility, dtype: int64
```

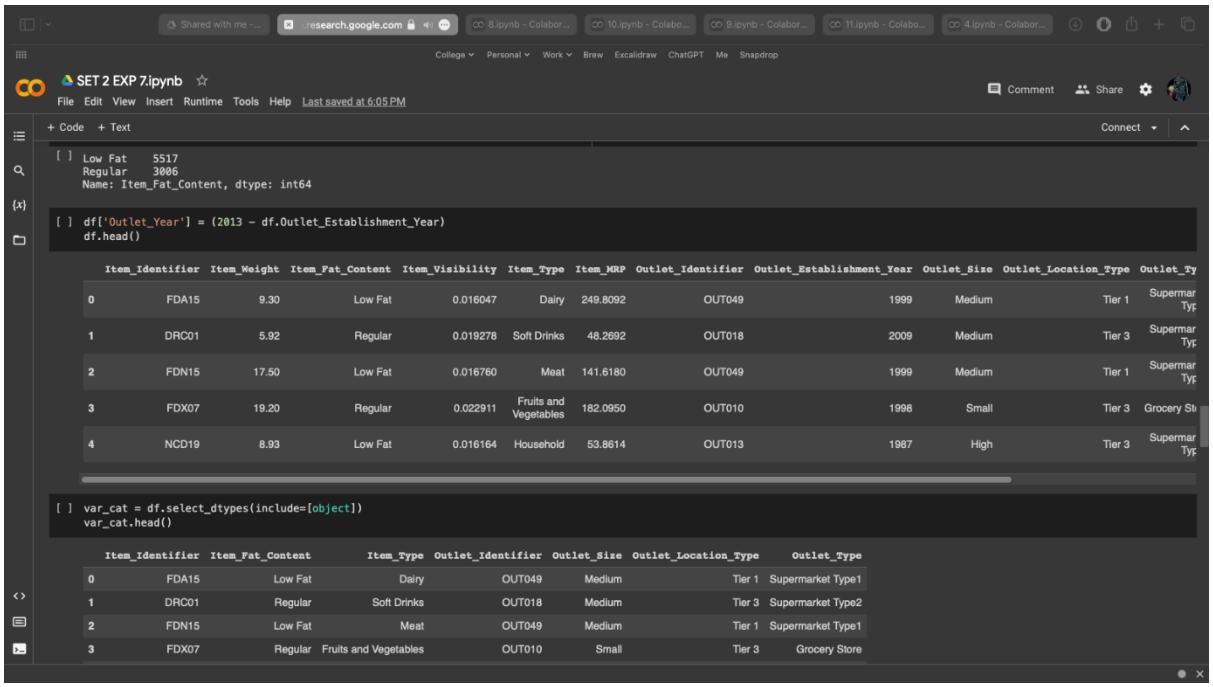

SET 2 EXP 7.ipynb
File Edit View Insert Runtime Tools Help Last saved at 6:05 PM
Comment
Share
Connect
+ Code + Text
0.073562 2
Name: Item_Visibility, dtype: int64
[ ] df.Outlet_Size.value_counts()
Medium 2793
Small 2388
High 932
Name: Outlet_Size, dtype: int64
[ ] crosstable = pd.crosstab(df['Outlet_Size'],df['Outlet_Type'])
crosstable
Outlet_Type Grocery Store Supermarket Type1 Supermarket Type2 Supermarket Type3
Outlet_Size
High 0 932 0 0
Medium 0 930 928 935
Small 528 1860 0 0
[ ] dic = {'Grocery Store':'Small'}
s = df.Outlet_Type.map(dic)
df.Outlet_Size= df.Outlet_Size.combine_first(s)
df.Outlet_Size.value_counts()
Small 2943
Medium 2793
High 932
Name: Outlet_Size, dtype: int64
[ ] # Checking if imputation was successful
df.isnull().sum(axis=0)

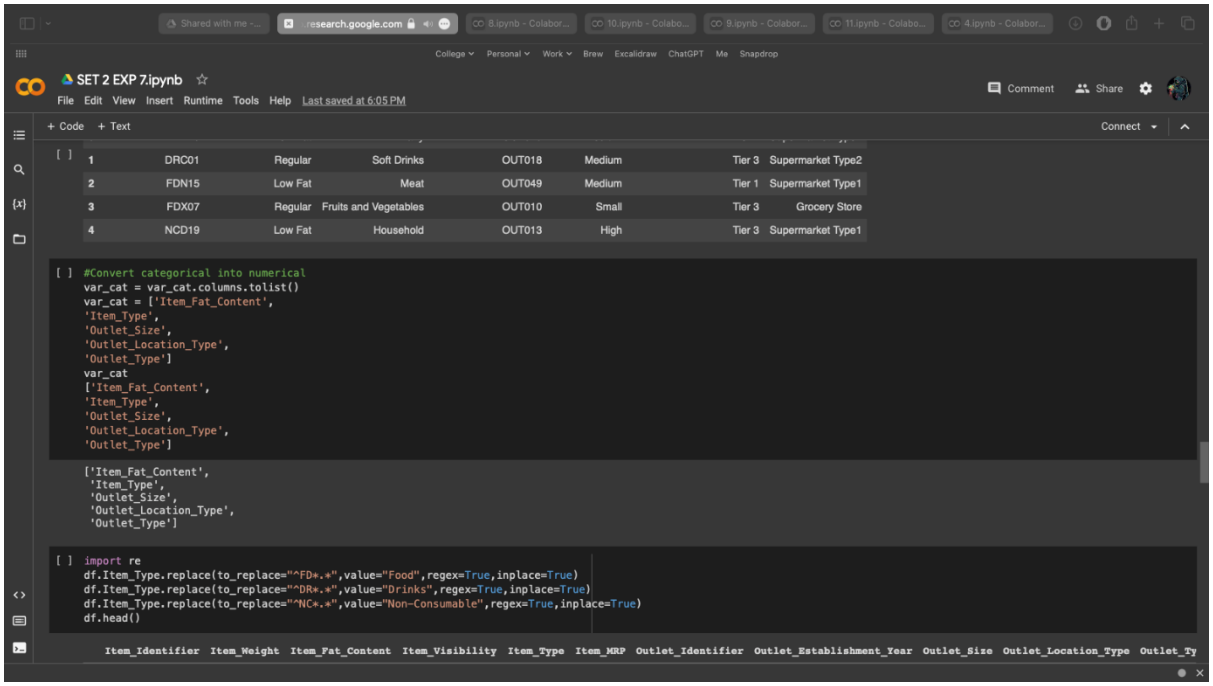

SET 2 EXP 7.ipynb
File Edit View Insert Runtime Tools Help Last saved at 6:05 PM
Comment
Share
Connect
+ Code + Text
[ ] df.isnull().sum(axis=0)
Item_Identifier 0
Item_Weight 1463
Item_Fat_Content 0
Item_Visibility 0
Item_Type 0
Item_MRP 0
Outlet_Identifier 0
Outlet_Establishment_Year 0
Outlet_Size 1855
Outlet_Location_Type 0
Outlet_Type 0
Item_Outlet_Sales 0
dtype: int64
[ ] dic = {"Tier 2":"Small"}
s = df.Outlet_Location_Type.map(dic)
df.Outlet_Size = df.Outlet_Size.combine_first(s)
[ ] df.isnull().sum(axis=0)
Item_Identifier 0
Item_Weight 1463
Item_Fat_Content 0
Item_Visibility 0
Item_Type 0
Item_MRP 0
Outlet_Identifier 0
Outlet_Establishment_Year 0
Outlet_Size 0
Outlet_Location_Type 0
Outlet_Type 0
Item_Outlet_Sales 0
dtype: int64
[ ] df['Item_Weight']=df['Item_Weight'].fillna(df.groupby('Item_Identifier')['Item_Weight'].transform('mean'))


SET 2 EXP 7.ipynb
File Edit View Insert Runtime Tools Help Last saved at 6:05 PM
Comment
Share
Connect
+ Code + Text
Item_Outlet_Sales 0
dtype: int64
[ ] df[df.Item_Weight.isnull()]
Item_Identifier Item_Weight Item_Fat_Content Item_Visibility Item_Type Item_MRP Outlet_Identifier Outlet_Establishment_Year Outlet_Size Outlet_Location_Type Outlet
927 FDN52 NaN Regular 0.130933 Frozen Foods 86.9198 OUT027 1985 Medium Tier 3 Super
1922 FDK57 NaN Low Fat 0.079904 Snack Foods 120.0440 OUT027 1985 Medium Tier 3 Super
4187 FDE52 NaN Regular 0.029742 Dairy 88.9514 OUT027 1985 Medium Tier 3 Super
5022 FDQ60 NaN Regular 0.191501 Baking Goods 121.2098 OUT019 1985 Small Tier 1 Grocer
[ ] # List of item types
item_type_list = df.Item_Type.unique().tolist()
# grouping based on item type and calculating mean of item weight
Item_Type_Means = df.groupby('Item_Type')['Item_Weight'].mean()
# Mapiing Item weight to item type mean
for i in item_type_list:
dic = {i:Item_Type_Means[i]}
s = df.Item_Type.map(dic)
df.Item_Weight = df.Item_Weight.combine_first(s)
Item_Type_Means = df.groupby('Item_Type')['Item_Weight'].mean()
# Checking if Imputation was successful
df.isnull().sum()
Item_Identifier 0
Item_Weight 0
Item_Fat_Content 0


SET 2 EXP 7.ipynb
File Edit View Insert Runtime Tools Help Last saved at 6:05 PM
Comment
Share
Connect
+ Code + Text
[ ] # List of item types
item_type_list = df.Item_Type.unique().tolist()
# grouping based on item type and calculating mean of item weight
Item_Type_Means = df.groupby('Item_Type')['Item_Weight'].mean()
# Mapiing Item weight to item type mean
for i in item_type_list:
dic = {i:Item_Type_Means[i]}
s = df.Item_Type.map(dic)
df.Item_Weight = df.Item_Weight.combine_first(s)
Item_Type_Means = df.groupby('Item_Type')['Item_Weight'].mean()
# Checking if Imputation was successful
df.isnull().sum()
Item_Identifier 0
Item_Weight 0
Item_Fat_Content 0
Item_Visibility 0
Item_Type 0
Item_MRP 0
Outlet_Identifier 0
Outlet_Establishment_Year 0
Outlet_Size 0
Outlet_Location_Type 0
Outlet_Type 0
Item_Outlet_Sales 0
dtype: int64
[ ] # Replacing 0's with NaN
import numpy as np
df.Item_Visibility.replace(to_replace=0.000000,value=np.NaN,inplace=True)
# Now fill by mean of visbility based on item identifiers
df.Item_Visibility = df.Item_Visibility.fillna(df.groupby('Item_Identifier')['Item_Visibility'].transform('mean'))
# Checking if Imputation was carried out successfully
df.isnull().sum()


SET 2 EXP 7.ipynb
File Edit View Insert Runtime Tools Help Last saved at 6:05 PM
Comment
Share
Connect
+ Code + Text
Item_Identifier 0
Item_Weight 0
Item_Fat_Content 0
Item_Visibility 0
Item_Type 0
Item_MRP 0
Outlet_Identifier 0
Outlet_Establishment_Year 0
Outlet_Size 0
Outlet_Location_Type 0
Outlet_Type 0
Item_Outlet_Sales 0
dtype: int64
df.Item_Fat_Content.value_counts()
Low Fat 5089
Regular 2889
LF 316
reg 117
low fat 112
Name: Item_Fat_Content, dtype: int64
df.Item_Fat_Content.replace(to_replace=["LF","low fat"],value="Low Fat",inplace=True)
df.Item_Fat_Content.replace(to_replace="reg",value="Regular",inplace=True)
df.Item_Fat_Content.value_counts()
Low Fat 5517
Regular 3006
Name: Item_Fat_Content, dtype: int64
df['Outlet_Year'] = (2013 - df.Outlet_Establishment_Year)
df.head()
Item_Identifier Item_Weight Item_Fat_Content Item_Visibility Item_Type Item_MRP Outlet_Identifier Outlet_Establishment_Year Outlet_Size Outlet_Location_Type Outlet_Ty


SET 2 EXP 7.ipynb
File Edit View Insert Runtime Tools Help Last saved at 6:05 PM
Comment
Share
Connect
+ Code + Text
Low Fat 5517
Regular 3006
Name: Item_Fat_Content, dtype: int64
df['Outlet_Year'] = (2013 - df.Outlet_Establishment_Year)
df.head()
Item_Identifier Item_Weight Item_Fat_Content Item_Visibility Item_Type Item_MRP Outlet_Identifier Outlet_Establishment_Year Outlet_Size Outlet_Location_Type Outlet_Ty
0 FDA15 9.30 Low Fat 0.016047 Dairy 249.8092 OUT049 1999 Medium Tier 1 Supermar Typ
1 DRC01 5.92 Regular 0.019278 Soft Drinks 48.2692 OUT018 2009 Medium Tier 3 Supermar Typ
2 FDN15 17.50 Low Fat 0.016760 Meat 141.6180 OUT049 1999 Medium Tier 1 Supermar Typ
3 FDX07 19.20 Regular 0.022911 Fruits and Vegetables 182.0950 OUT010 1998 Small Tier 3 Grocery St
4 NCD19 8.93 Low Fat 0.016164 Household 53.8614 OUT013 1987 High Tier 3 Supermar Typ
var_cat = df.select_dtypes(include=[object])
var_cat.head()
Item_Identifier Item_Fat_Content Item_Type Outlet_Identifier Outlet_Size Outlet_Location_Type Outlet_Type
0 FDA15 Low Fat Dairy OUT049 Medium Tier 1 Supermarket Type1
1 DRC01 Regular Soft Drinks OUT018 Medium Tier 3 Supermarket Type2
2 FDN15 Low Fat Meat OUT049 Medium Tier 1 Supermarket Type1
3 FDX07 Regular Fruits and Vegetables OUT010 Small Tier 3 Grocery Store



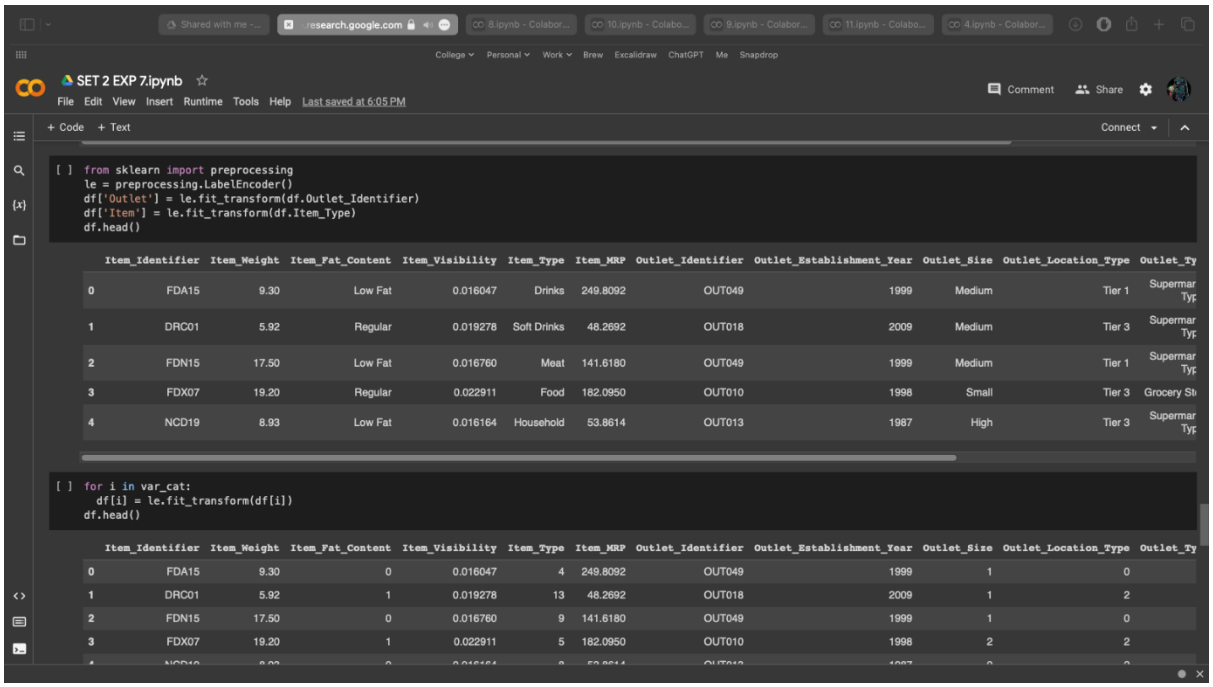
1	DRC01	Regular	Soft Drinks	OUT018	Medium	Tier 3	Supermarket Type2
2	FDN15	Low Fat	Meat	OUT049	Medium	Tier 1	Supermarket Type1
3	FDX07	Regular	Fruits and Vegetables	OUT010	Small	Tier 3	Grocery Store
4	NCD19	Low Fat	Household	OUT013	High	Tier 3	Supermarket Type1

```
#Convert categorical into numerical
var_cat = var_cat.columns.tolist()
var_cat = ['Item_Fat_Content',
'Item_Type',
'Outlet_Size',
'Outlet_Location_Type',
'Outlet_Type']
var_cat
['Item_Fat_Content',
'Item_Type',
'Outlet_Size',
'Outlet_Location_Type',
'Outlet_Type']
```

```
['Item_Fat_Content',
 'Item_Type',
 'Outlet_Size',
 'Outlet_Location_Type',
 'Outlet_Type']
```

```
import re
df.Item_Type.replace(to_replace="^FD*.*",value="Food",regex=True,inplace=True)
df.Item_Type.replace(to_replace="^DR*.*",value="Drinks",regex=True,inplace=True)
df.Item_Type.replace(to_replace="^NC*.*",value="Non-Consumable",regex=True,inplace=True)
df.head()
```

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Outlet_Size	Outlet_Location_Type	Outlet_Ty

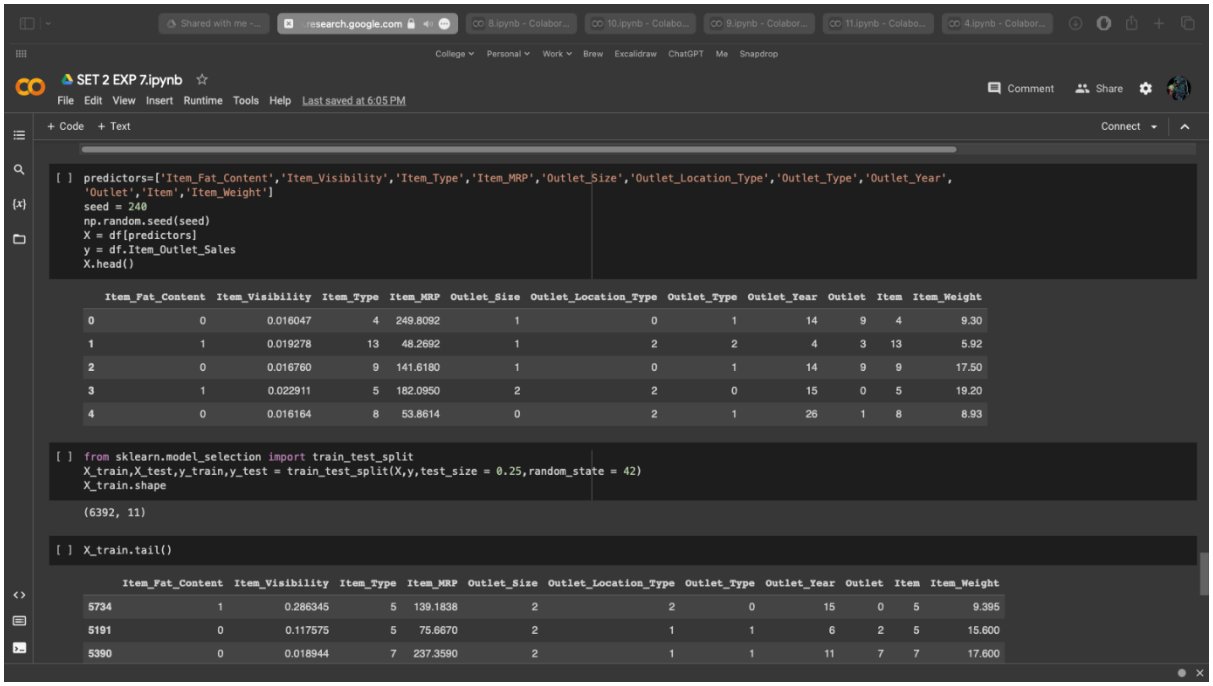


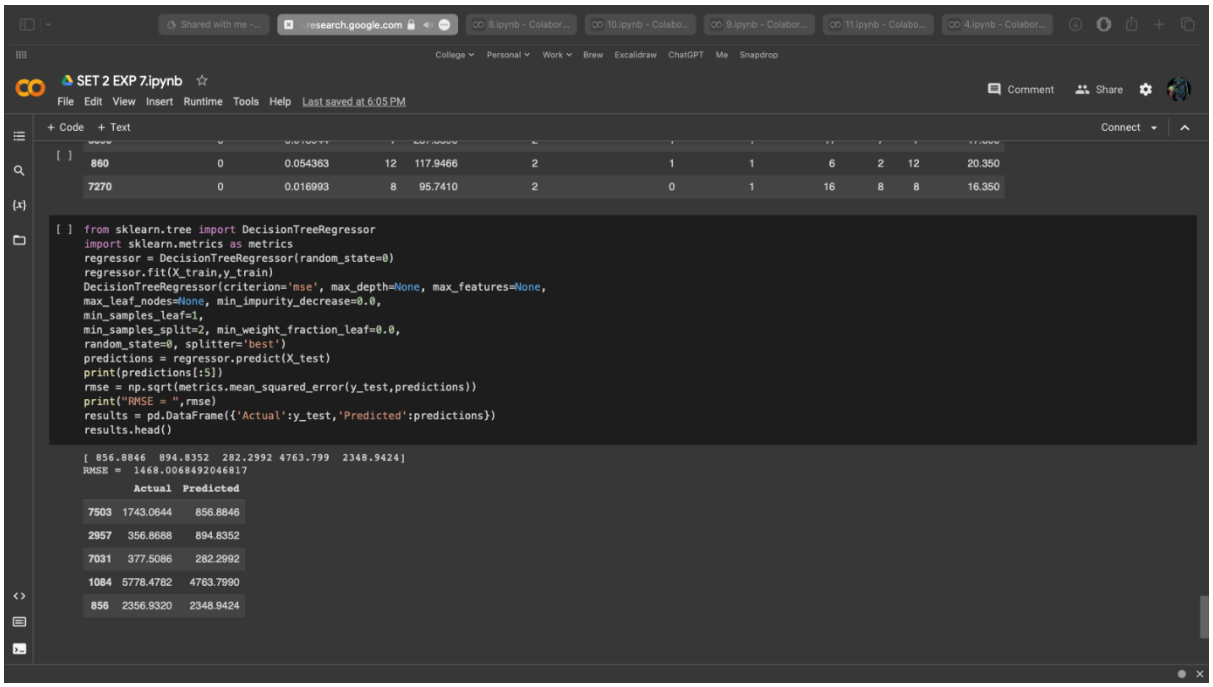

```
from sklearn import preprocessing
le = preprocessing.LabelEncoder()
df['Outlet'] = le.fit_transform(df.Outlet_Identifier)
df['Item'] = le.fit_transform(df.Item_Type)
df.head()
```

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Outlet_Size	Outlet_Location_Type	Outlet_Ty
0	FDA15	9.30	Low Fat	0.016047	Drinks	249.8092	OUT049	1999	Medium	Tier 1	Supermar Typ
1	DRC01	5.92	Regular	0.019278	Soft Drinks	48.2692	OUT018	2009	Medium	Tier 3	Supermar Typ
2	FDN15	17.50	Low Fat	0.016760	Meat	141.6180	OUT049	1999	Medium	Tier 1	Supermar Typ
3	FDX07	19.20	Regular	0.022911	Food	182.0950	OUT010	1998	Small	Tier 3	Grocery St
4	NCD19	8.93	Low Fat	0.016164	Household	53.8614	OUT013	1987	High	Tier 3	Supermar Typ

```
for i in var_cat:
  df[i] = le.fit_transform(df[i])
df.head()
```

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Outlet_Size	Outlet_Location_Type	Outlet_Ty
0	FDA15	9.30	0	0.016047	4	249.8092	OUT049	1999	1	0	
1	DRC01	5.92	1	0.019278	13	48.2692	OUT018	2009	1	2	
2	FDN15	17.50	0	0.016760	9	141.6180	OUT049	1999	1	0	
3	FDX07	19.20	1	0.022911	5	182.0950	OUT010	1998	2	2	





RESULT

Thus, the machine learning - regression in python using the Decision tree classifier has been implemented and executed successfully.

EXP. NO: 8 **DATE:**

VARIOUS REGRESSION ALGORITHM ANALYSIS

AIM

To analyze various Regression algorithms in machine learning using python.

ALGORITHM

1. Import the required libraries including "drive" and "pandas".
2. Mount the Google Drive using the "drive.mount" function.
3. Read the "boston_house_prices.csv" dataset using the "pd.read_csv" function and store it in "df".
4. Display the first few rows of the dataset using "df.head()".
5. Create a new dataframe "d1" containing only the columns "LSTAT" and "MEDV" using the "df.loc" function and store it in "d1".
6. Display the first few rows of "d1" using "d1.head()".
7. Visualize the dataset by creating a scatter plot using the "df.plot" function with "LSTAT" on the x-axis and "MEDV" on the y-axis. Label the axes using "plt.xlabel" and "plt.ylabel" functions respectively. Display the plot using "plt.show()".
8. Create two dataframes "x" and "y" containing the "LSTAT" and "MEDV" columns respectively.
9. Split the dataset into training and testing sets using the "train_test_split" function from the "sklearn.model_selection" module with test size of 0.2 and random state of 1. Store the split data in "x_train", "x_test", "y_train", and "y_test".
10. Fit a linear regression model to the training data using "LinearRegression" function from "sklearn.linear_model" module. Predict the target values for the testing data using the "predict" method and store the predictions in "y_pred".

CODE SEGMENT

```
from google.colab import drive
drive.mount('/content/drive')

import pandas as pd
df=pd.read_csv("/content/drive/My Drive/Lab Programs/Ex 8/boston_house_prices.csv")

df.head()

d1=df.loc[:,["LSTAT","MEDV"]]
d1.head()

import matplotlib.pyplot as plt
df.plot(x="LSTAT",y="MEDV",style='o')
plt.xlabel("LSTAT")
plt.ylabel("MEDV")
```

```
plt.show()

x=pd.DataFrame(df["LSTAT"])
y=pd.DataFrame(df["MEDV"])
print(y)

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=1)

print(x_train.shape,x_test.shape,y_train.shape,y_test.shape)

from sklearn.linear_model import LinearRegression
reg=LinearRegression()
reg.fit(x_train,y_train)
y_pred=reg.predict(x_test)
print(reg.intercept_)
print(reg.coef_)

print(len(y_pred),len(y_test))

from sklearn import metrics
import numpy as np
print("\nMean Absolute Error ",metrics.mean_absolute_error(y_pred,y_test))
print("\nMean Squared Error ",metrics.mean_squared_error(y_pred,y_test))
print("\nRoot Mean Squared Error ",np.sqrt(metrics.mean_squared_error(y_pred,y_test)))
```

INPUT / OUTPUT

SET 1 EXP 8.ipynb

File Edit View Insert Runtime Tools Help Last saved at 6:29 PM

+ Code + Text

```
import pandas as pd
df=pd.read_csv("boston_house_prices.csv")
```

```
[ ] df.head()
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	5.33	36.2

```
d1=df.loc[:,["LSTAT","MEDV"]]
d1.head()
```

	LSTAT	MEDV
0	4.98	24.0
1	9.14	21.6
2	4.03	34.7
3	2.94	33.4
4	5.33	36.2

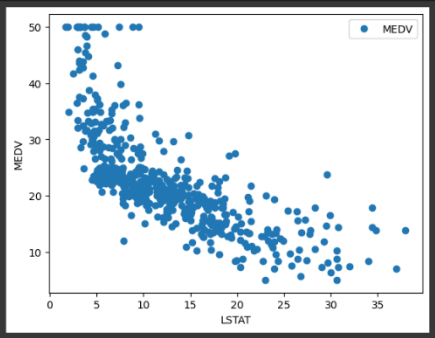
```
[ ] import matplotlib.pyplot as plt
    df.plot(x="LSTAT",y="MEDV",style='o')
    plt.xlabel("LSTAT")
```

SET 1 EXP 8.ipynb

File Edit View Insert Runtime Tools Help Last saved at 6:29 PM

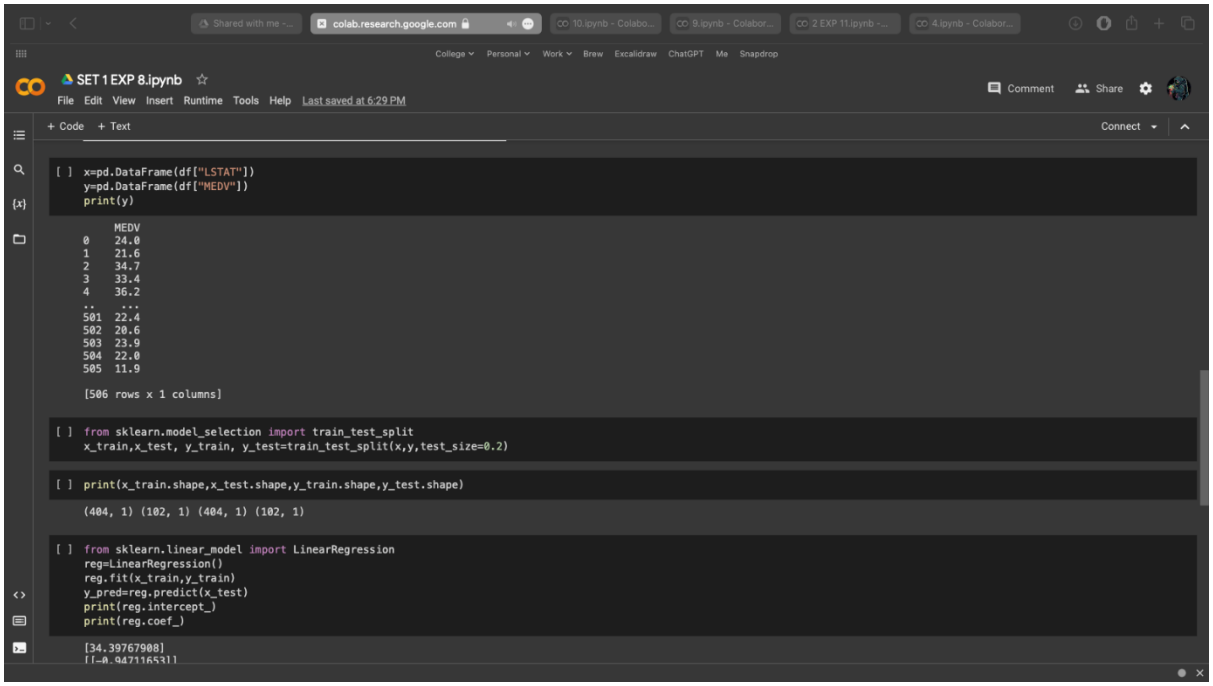
+ Code + Text

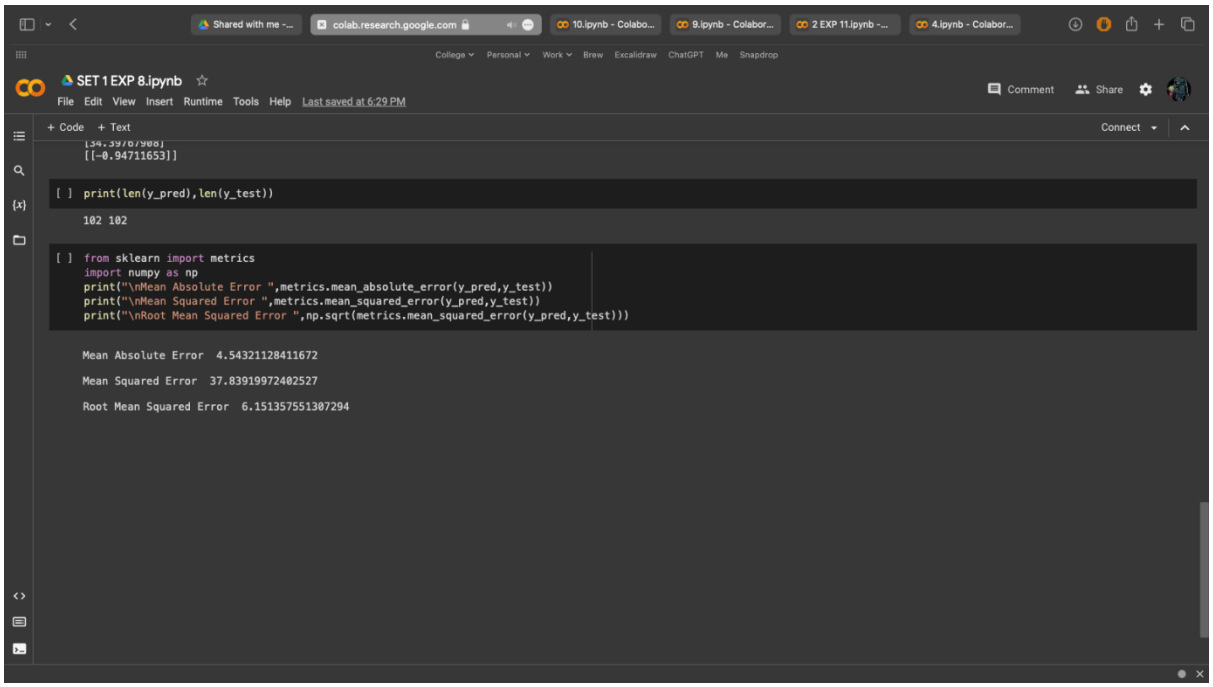
```
[ ] import matplotlib.pyplot as plt
    df.plot(x="LSTAT",y="MEDV",style='o')
    plt.xlabel("LSTAT")
    plt.ylabel("MEDV")
    plt.show()
```




```
[ ] x=pd.DataFrame(df["LSTAT"])
    y=pd.DataFrame(df["MEDV"])
    print(y)
```

```
     MEDV
0    24.0
```





RESULT

Thus, the machine learning - regression algorithms using python have been analyzed and implemented successfully.

EXP. NO: 9 **DATE:**

UNSUPERVISED LEARNING - AGGLOMERATIVE, BIRCH, DBSCAN

AIM

To implement various unsupervised machine learning algorithms such as Agglomerative, BIRCH, DBSCAN using python.

ALGORITHM

1. Import the scikit-learn library and print the version.
2. Generate a synthetic classification dataset using the make_classification() function from scikit-learn. Store the input features in X and the corresponding class labels in y.
3. Plot the dataset using a scatter plot where the class labels are represented by different colors.
4. Perform agglomerative clustering on the dataset using the AgglomerativeClustering() function from scikit-learn. Set the number of clusters to 2.
5. Fit the model to the dataset and obtain the predicted cluster labels using the fit_predict() method.
6. Obtain the unique cluster labels using the unique() function from NumPy.
7. Plot the dataset using a scatter plot where the predicted cluster labels are represented by different colors.
8. Perform DBSCAN clustering on the dataset using the DBSCAN() function from scikit-learn. Set the value of epsilon to 0.3 and the minimum number of samples to 9.
9. Fit the model to the dataset and obtain the predicted cluster labels using the fit_predict() method.
10. Plot the dataset using a scatter plot where the predicted cluster labels are represented by different colors.

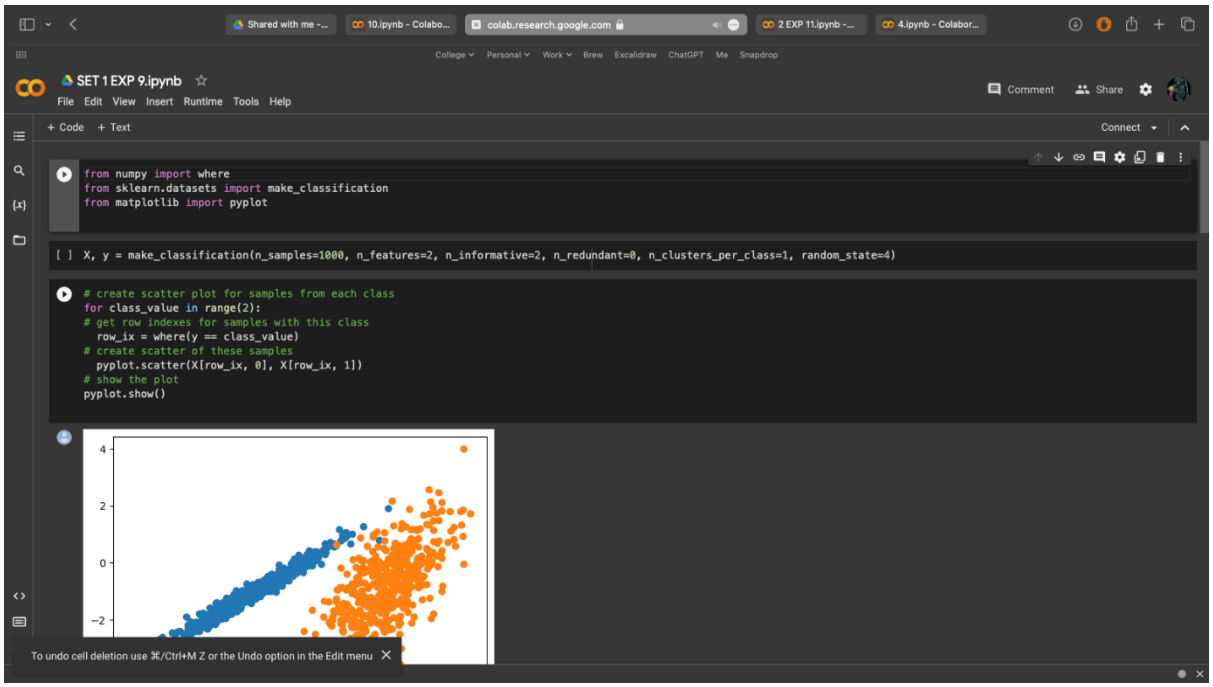
CODE SEGMENT

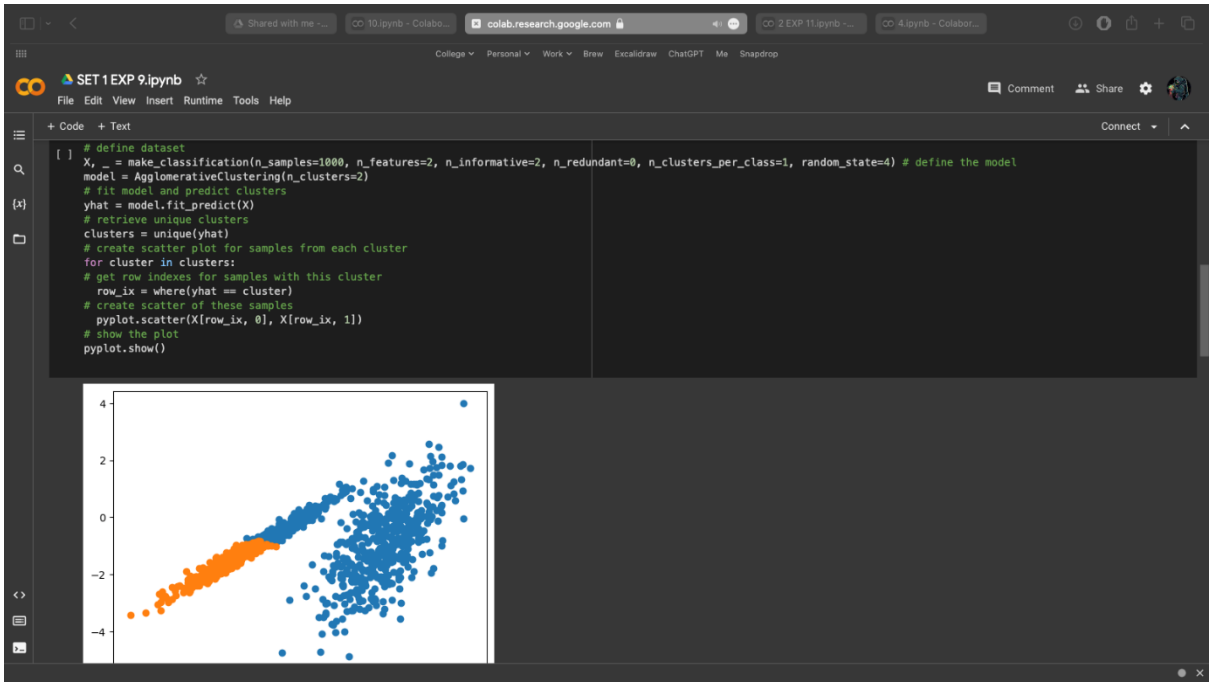
```
import sklearn
print(sklearn.__version__)

# synthetic classification dataset
from numpy import where
from sklearn.datasets import make_classification
from matplotlib import pyplot
# define dataset
X, y = make_classification(n_samples=1000, n_features=2, n_informative=2, n_redundant=0,
n_clusters_per_class=1, random_state=4)
# create scatter plot for samples from each class
for class_value in range(2):
# get row indexes for samples with this class
```

```
    row_ix = where(y == class_value)
# create scatter of these samples
    pyplot.scatter(X[row_ix, 0], X[row_ix, 1])
# show the plot
pyplot.show()

# agglomerative clustering
from numpy import unique
from numpy import where
from sklearn.datasets import make_classification
from sklearn.cluster import AgglomerativeClustering
from matplotlib import pyplot
# define dataset
X, _ = make_classification(n_samples=1000, n_features=2, n_informative=2, n_redundant=0,
n_clusters_per_class=1, random_state=4)
# define the model
model = AgglomerativeClustering(n_clusters=2)
# fit model and predict clusters
yhat = model.fit_predict(X)
# retrieve unique clusters
clusters = unique(yhat)
# create scatter plot for samples from each cluster
for cluster in clusters:
# get row indexes for samples with this cluster
    row_ix = where(yhat == cluster)
# create scatter of these samples
    pyplot.scatter(X[row_ix, 0], X[row_ix, 1])
# show the plot
pyplot.show()

# dbscan clustering
from numpy import unique
from numpy import where
from sklearn.datasets import make_classification
from sklearn.cluster import DBSCAN
from matplotlib import pyplot
# define dataset
X, _ = make_classification(n_samples=1000, n_features=2, n_informative=2, n_redundant=0,
n_clusters_per_class=1, random_state=4)
# define the model
model = DBSCAN(eps=0.30, min_samples=9)
# fit model and predict clusters
yhat = model.fit_predict(X)
# retrieve unique clusters
clusters = unique(yhat)
# create scatter plot for samples from each cluster
```

```
for cluster in clusters:
# get row indexes for samples with this cluster
    row_ix = where(yhat == cluster)
# create scatter of these samples
    pyplot.scatter(X[row_ix, 0], X[row_ix, 1])
# show the plot
pyplot.show()
```

INPUT / OUTPUT



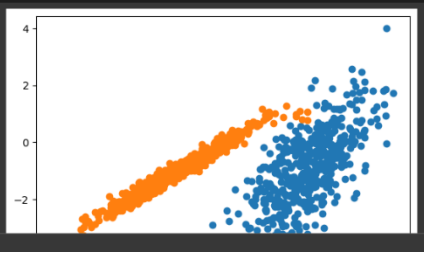


SET 1 EXP 9.ipynb

File Edit View Insert Runtime Tools Help

+ Code + Text

```
[ ] from sklearn.cluster import Birch
    from matplotlib import pyplot
    # define dataset
    X, _ = make_classification(n_samples=1000, n_features=2, n_informative=2, n_redundant=0, n_clusters_per_class=1, random_state=4) # define the model
    model = Birch(threshold=0.01, n_clusters=2)
    # fit the model
    model.fit(X)
    # assign a cluster to each example
    yhat = model.predict(X)
    # retrieve unique clusters
    clusters = unique(yhat)
    # create scatter plot for samples from each cluster
    for cluster in clusters:
    # get row indexes for samples with this cluster
      row_ix = where(yhat == cluster)
    # create scatter of these samples
      pyplot.scatter(X[row_ix, 0], X[row_ix, 1])
    # show the plot
    pyplot.show()
```

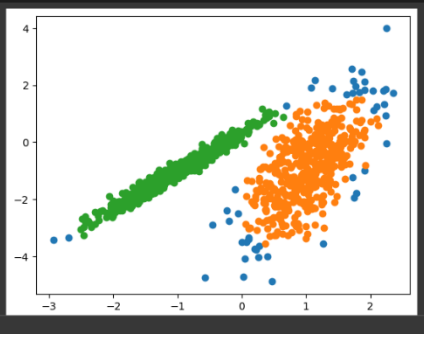


SET 1 EXP 9.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
    model = DBSCAN(eps=0.30, min_samples=9)
    # fit model and predict clusters
    yhat = model.fit_predict(X)
    # retrieve unique clusters
    clusters = unique(yhat)
    # create scatter plot for samples from each cluster
    for cluster in clusters:
    # get row indexes for samples with this cluster
      row_ix = where(yhat == cluster)
    # create scatter of these samples
      pyplot.scatter(X[row_ix, 0], X[row_ix, 1])
    # show the plot
    pyplot.show()
```



RESULT
Thus, the various unsupervised machine learning algorithms have been implemented and executed successfully

EXP. NO: 10 **DATE:**

K-MEANS CLUSTERING WITH ELBOW METHOD

AIM

To implement K-means clustering with elbow method in machine learning using python.

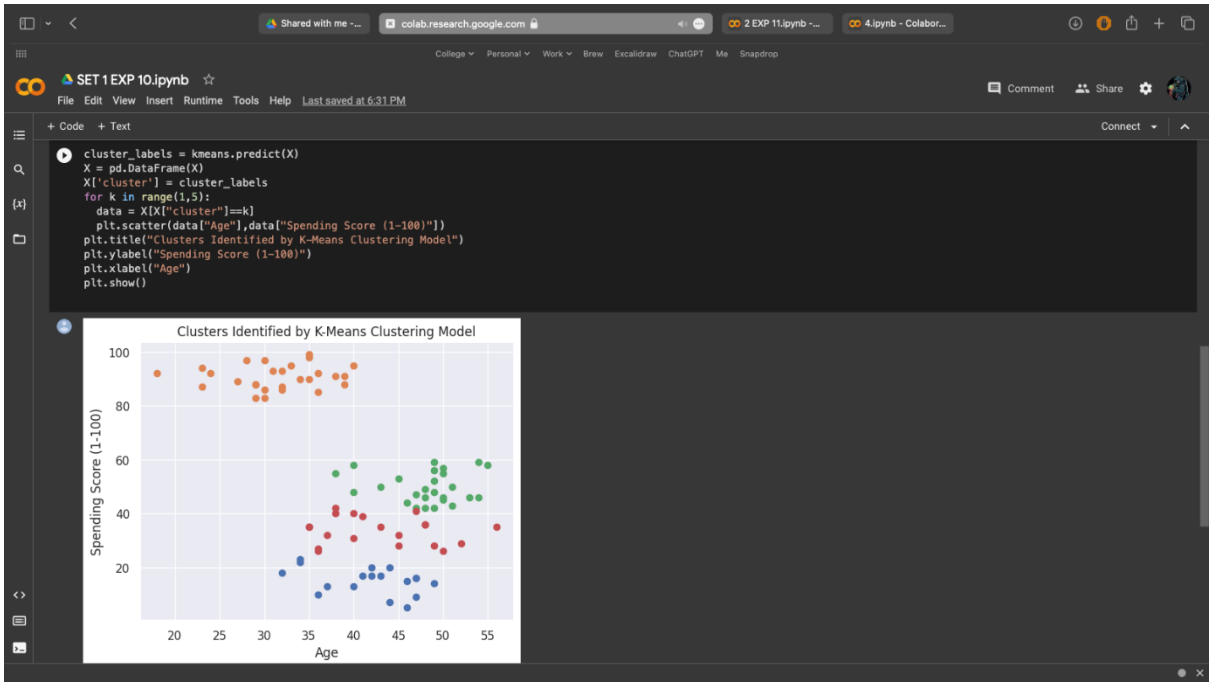
ALGORITHM

1. Import the required libraries including pandas, matplotlib, seaborn and KMeans from sklearn.cluster.
2. Load the dataset from the CSV file using pd.read_csv() function and store it in a variable named "df".
3. Prepare the data by creating a new dataframe named "X" containing the required columns, 'Age' and 'Spending Score (1-100)'.
4. Perform KMeans clustering on the data by iterating over a range of values of 'k' from 1 to 10 and storing the sum of squared distances of samples to their closest cluster center in a list ..
5. Finally, plot the clusters identified by the KMeans model by assigning the predicted cluster labels to the data points and plotting them using scatter plot using the 'Age' and 'Spending Score (1-100)' columns.

CODE SEGMENT

```
import pandas as pd
df = pd.read_csv("Mall_Customers.csv")
print(df.head())

from sklearn.cluster import KMeans

X = df[['Age', 'Spending Score (1-100)']].copy()

wcss=[]
for i in range(1,11):
    kmeans = KMeans(init="k-means++",n_clusters=i, n_init=100, random_state=0)
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)

import matplotlib.pyplot as plt
import seaborn as sns
sns.set()
plt.plot(range(1, 11), wcss)
plt.title('Selecting the Numbeer of Clusters using the Elbow Method')
plt.xlabel('Clusters')
plt.ylabel('WCSS')
plt.show()

cluster_labels = kmeans.predict(X)
X = pd.DataFrame(X)
X['cluster'] = cluster_labels

for k in range(1,5):
    data = X[X["cluster"]==k]
    plt.scatter(data["Age"],data["Spending Score (1-100)"])
plt.title("Clusters Identified by K-Means Clustering Model")
plt.ylabel("Spending Score (1-100)")
plt.xlabel("Age")
plt.show()
```

INPUT / OUTPUT



RESULT

Thus, the K-means clustering with the elbow method in machine learning using python have been implemented and executed successfully.

EXP. NO: 11 **DATE:**

A-PRIORI ALGORITHM

AIM

To implement A-Priori algorithm in machine learning using python.

ALGORITHM

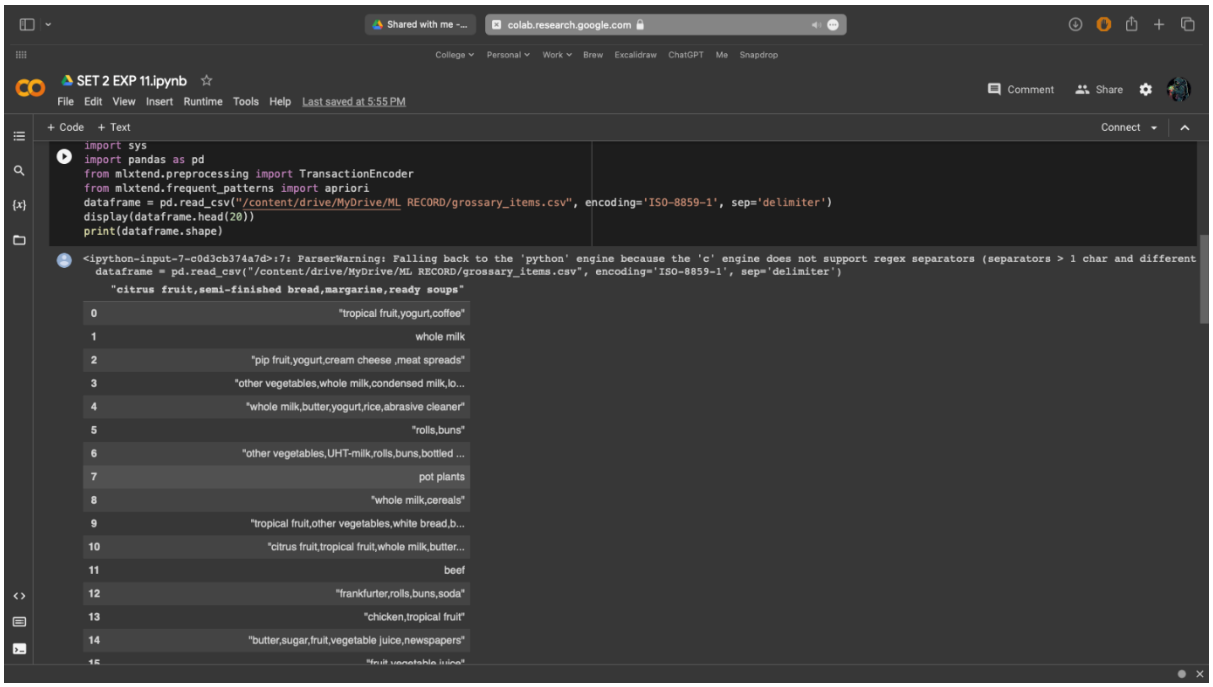
1. Import necessary libraries - sys, pandas, TransactionEncoder from mlxtend.preprocessing, apriori and association_rules from mlxtend.frequent_patterns.
2. Read the csv file "grossary_items.csv" using pandas and store it in the dataframe variable.
3. Display the first five rows of the dataframe using the display() function and print the shape of the dataframe using the shape attribute.
4. Convert the dataframe to a list and convert each string in the list to multiple strings separated by commas.
5. Encode the dataset using TransactionEncoder and store it in the te variable.
6. Transform the encoded dataset using the transform() method of te and store it in te_ary variable.
7. Convert the transformed array to a pandas dataframe using pd.DataFrame() and store it in the df variable.
8. Generate frequent itemsets using the Apriori algorithm by calling apriori() from mlxtend.frequent_patterns and passing df as input along with min_support and use_colnames parameters.
9. Generate association rules using association_rules() from mlxtend.frequent_patterns and pass the frequent_itemsets generated in the previous step along with the metric and min_threshold parameters.
10. Display the first five rows of the resulting dataframe that contains the antecedents, consequents and support of the association rules using the head() method.

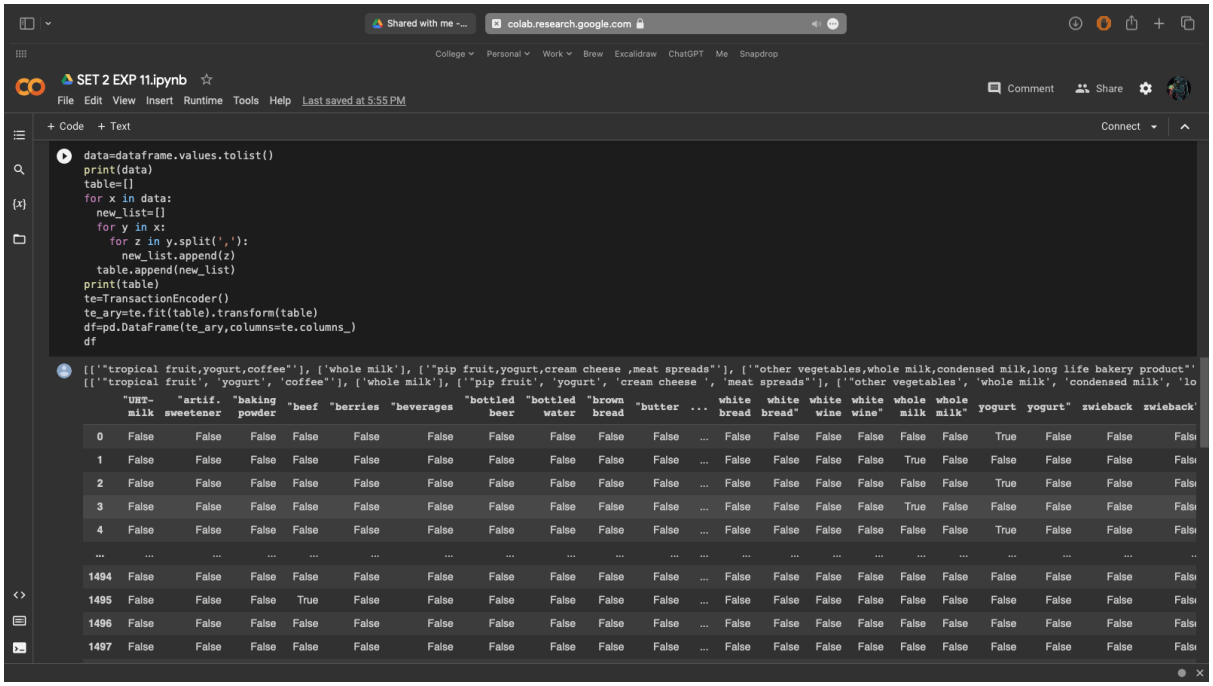
CODE SEGMENT

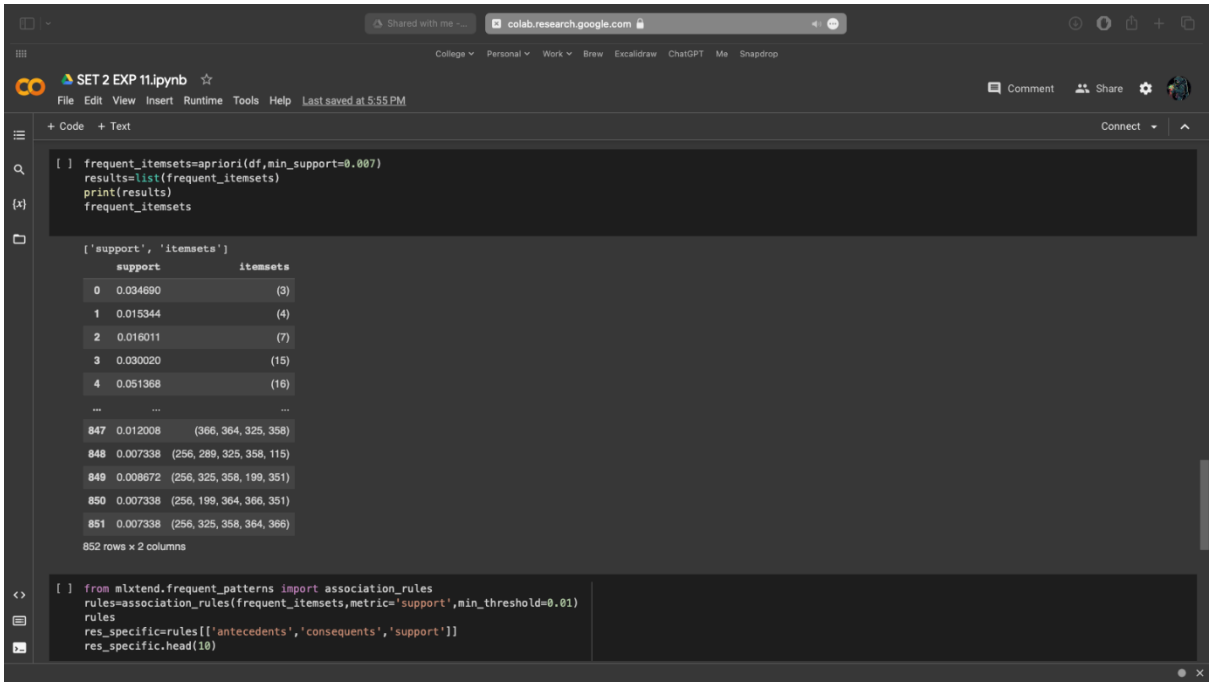
```
import sys
import pandas as pd
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import apriori

dataframe = pd.read_csv("grossary_items.csv")
display(dataframe.head())
print(dataframe.shape)

#converting the dataframe to a list
data = dataframe.values.tolist()
#convert the single string in each list to multiple strings separated by commas
table = []
for x in data:
```

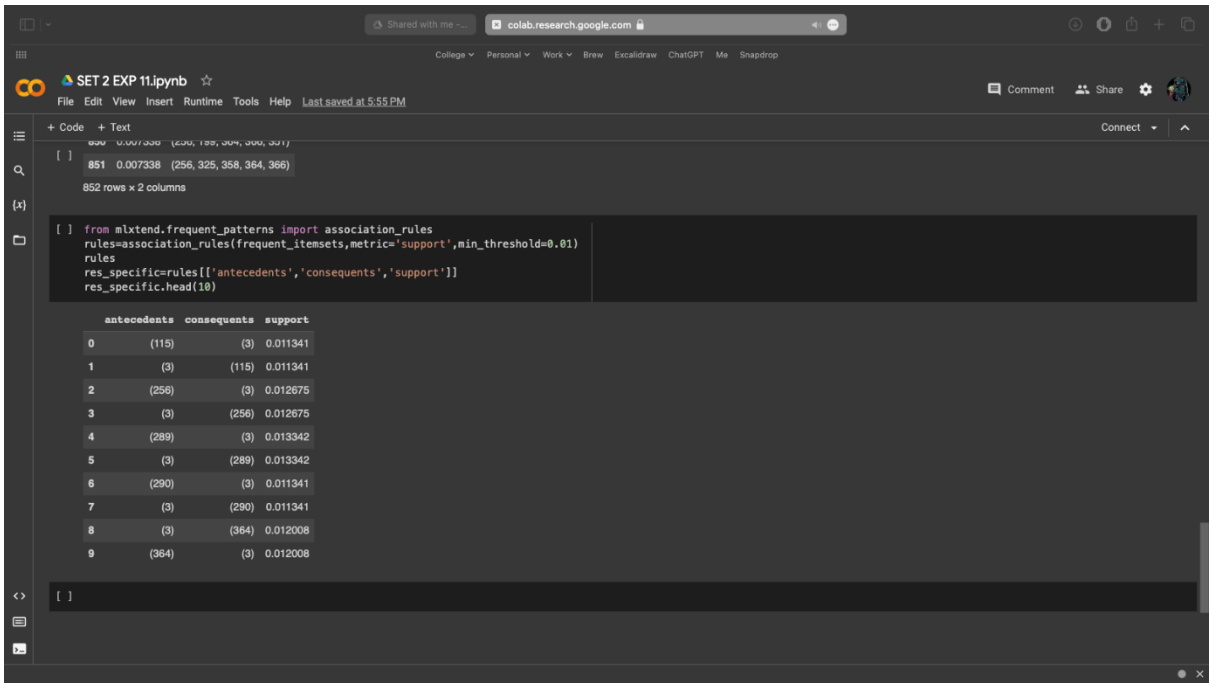
```
    new_list = []
    for y in x:
        for z in y.split(','):
            new_list.append(z)
    table.append(new_list)
#encode the dataset
te = TransactionEncoder()
te_ary = te.fit(table).transform(table)
df = pd.DataFrame(te_ary, columns=te.columns_)
df

#generate frequent itemsets using Apriori algorithm
#frequent_itemsets = apriori(df,min_support=0.1,use_colnames=True)
frequent_itemsets = apriori(df, min_support = 0.007)
# Visualising the results
results = list(frequent_itemsets)
print(results)
#print(df.head())
#print(df.shape)
frequent_itemsets

from mlxtend.frequent_patterns import association_rules
rules=association_rules(frequent_itemsets,metric='support',min_threshold=0.01)
rules
res_specific = rules[['antecedents', 'consequents', 'support']]
res_specific.head()
```

INPUT / OUTPUT









RESULT

Thus, the A-Priori algorithm in machine learning using python has been implemented and executed successfully.

EXP. NO: 12 **DATE:**

EMPLOYEE ATTRITION PREDICTION USING ORANGE DATA MINING

AIM

To implement the employee attrition prediction using orange data mining.

PROCEDURE

Step 1: Upload Dataset

- Click on File. This will put the widget File on the canvas.
- Double click File to open its properties tab.
- Click on the button to navigate to your csv files.
- Connect the widget to the file [Dataset]_Module8_Train(Employee).csv

Step 2: Clean Missing Data

- Clean any missing data from the csv file, which we have just uploaded in Step 1.
- Double-click on the widget TrainData to verify if there is any missing data.
- Insert the widget Feature Statistics into the canvas.
- Connect widget TrainData to widget Feature Statistics, by dragging the output from TrainData to the input of Feature Statistics.
- After the connection is made, double-click on Feature Statistics to see the results.
- Insert the widget Impute onto canvas.
- Connect widget TrainData to widget Impute, by dragging the output from TrainData to the input of Impute.
- After the connection is made, double-click on Impute to open up the properties tab.
- Connect the output of Impute to the input of the existing Feature Statistics to see the results.

Step 3: Edit Type

- In TrainData, Feature Type for some of the columns is Numeric Feature.
- Change their Feature Type from Numeric Feature to Categorical Feature.
- This can be done with the help of TestData widget.

Step 4: Select Columns

- Select the column Attrition_rate as our label (Output).
- Change the Feature Type for Attrition_rate, from Numeric Feature to Numeric Label.
- Use Select column widget to perform this task.
- To make Attrition_rate as the target, click and drag it into the Target Variable area.

Step 5: Data Sampler

- Split the TrainingData_Attrition.csv into both the training data and validation data.
- To do this, we can use Data Sampler.
- Connect widget Select Columns to widget Data Sampler.
- Set the ratio for training data and testing data. Normally it is 70:30 or 80:20.

Step 6: Train Model

- Train the model to find its own rules.
- Use the widget called Test and Score.
- For training purposes, we need to have 2 inputs, one for the training data and another for the learning algorithm.
- We can insert more than 1 learning algorithm if we want to do a comparison among them.
- For the training data, we will use the Data Sample from Data Sampler, which represents the 80% data.
- Connect widget Data Sampler to widget Test and Score.

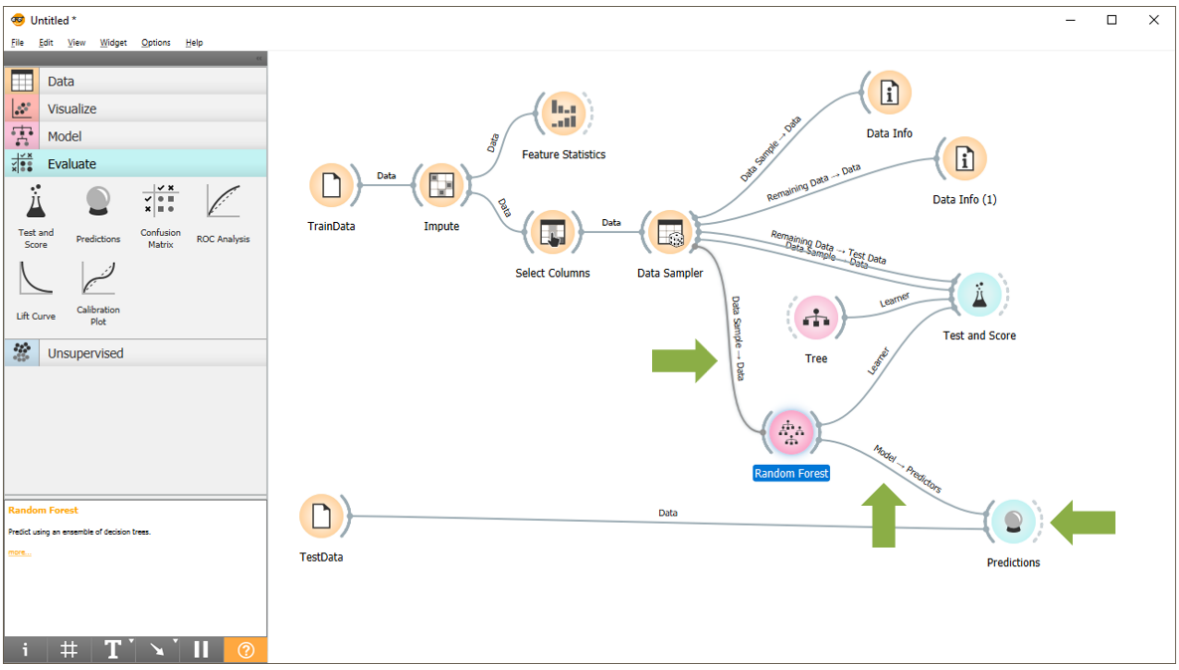
Step 7: Score Model

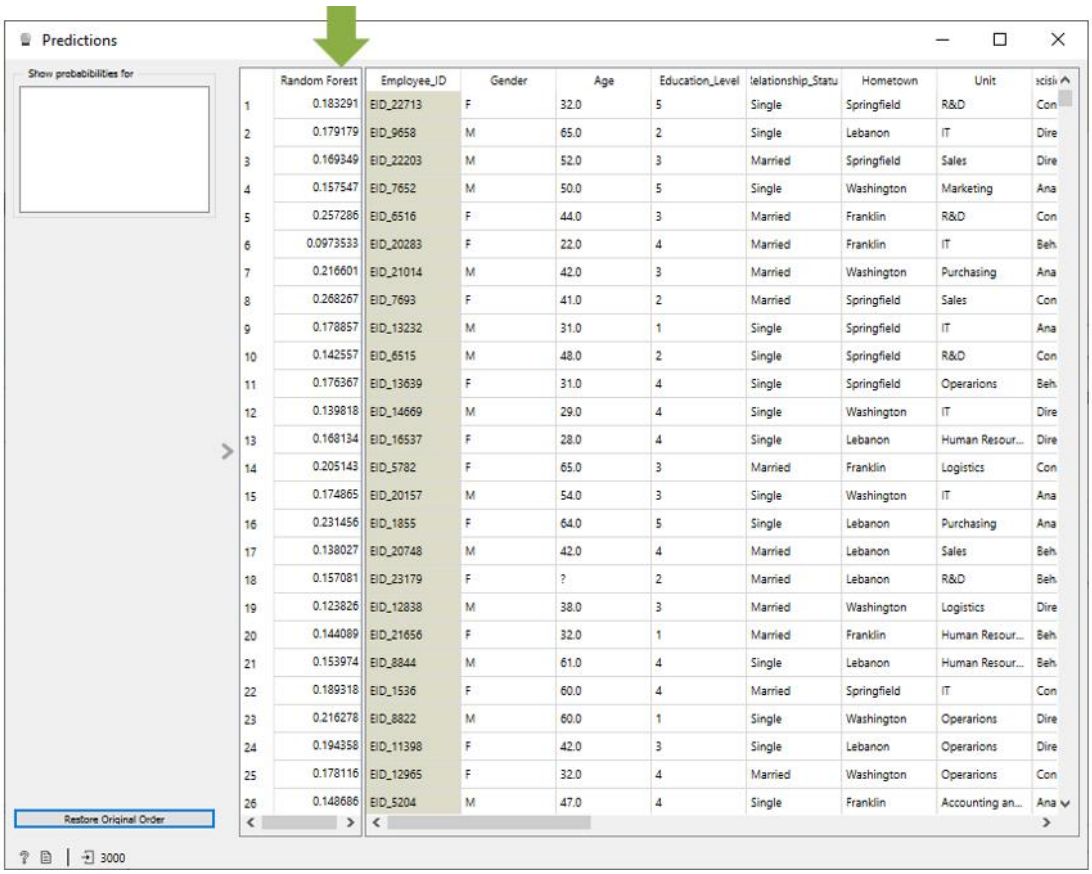
- To verify the performance of the model this is used.
- Since we will be using this to verify the performance, we will connect it to the Remaining Data of Data Sampler.

Step 8: Predictions

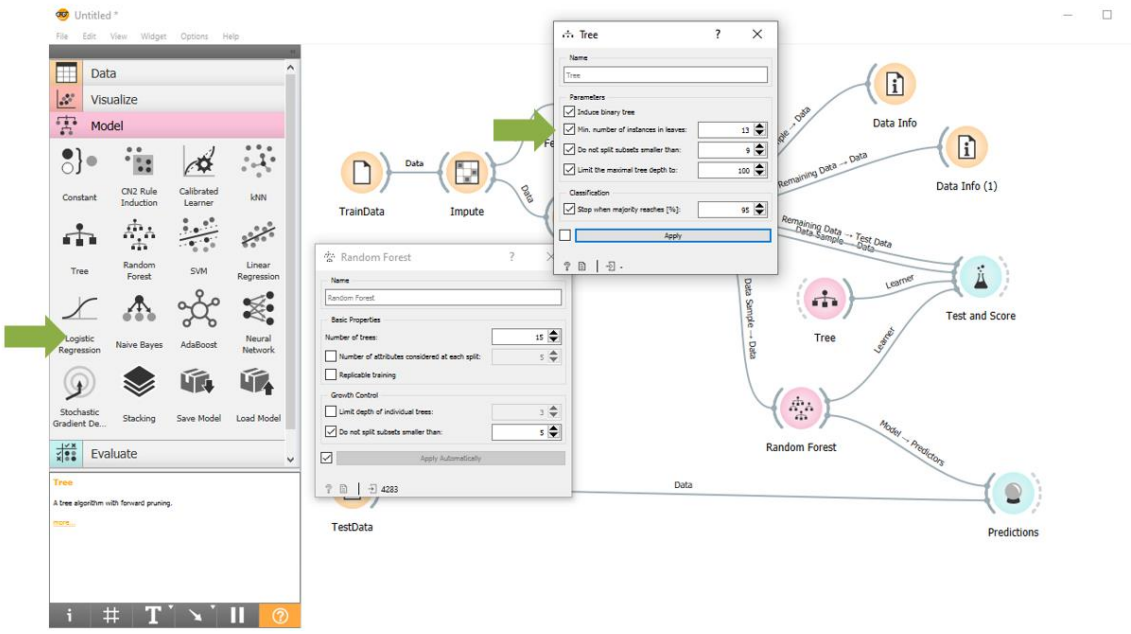
- Use the trained model to make predictions.
- Testing data will be the input to the prediction widget.

OUTPUT





	Random Forest	Employee_ID	Gender	Age	Education_Level	Relationship_Statu	Hometown	Unit	scisi
1	0.183291	EID_22713	F	32.0	5	Single	Springfield	R&D	Con
2	0.179179	EID_9658	M	65.0	2	Single	Lebanon	IT	Dire
3	0.169349	EID_22203	M	52.0	3	Married	Springfield	Sales	Dire
4	0.157547	EID_7652	M	50.0	5	Single	Washington	Marketing	Ana
5	0.257286	EID_6516	F	44.0	3	Married	Franklin	R&D	Con
6	0.0973533	EID_20283	F	22.0	4	Married	Franklin	IT	Beh
7	0.216601	EID_21014	M	42.0	3	Married	Washington	Purchasing	Ana
8	0.268267	EID_7693	F	41.0	2	Married	Springfield	Sales	Con
9	0.178857	EID_13232	M	31.0	1	Single	Springfield	IT	Ana
10	0.142557	EID_6515	M	48.0	2	Single	Springfield	R&D	Con
11	0.176367	EID_13639	F	31.0	4	Single	Springfield	Operarions	Beh
12	0.139818	EID_14669	M	29.0	4	Single	Washington	IT	Dire
13	0.168134	EID_16537	F	28.0	4	Single	Lebanon	Human Resour...	Dire
14	0.205143	EID_5782	F	65.0	3	Married	Franklin	Logistics	Con
15	0.174865	EID_20157	M	54.0	3	Single	Washington	IT	Ana
16	0.231456	EID_1855	F	64.0	5	Single	Lebanon	Purchasing	Ana
17	0.138027	EID_20748	M	42.0	4	Married	Lebanon	Sales	Beh
18	0.157081	EID_23179	F	?	2	Married	Lebanon	R&D	Beh
19	0.123826	EID_12838	M	38.0	3	Married	Washington	Logistics	Dire
20	0.144089	EID_21656	F	32.0	1	Married	Franklin	Human Resour...	Beh
21	0.153974	EID_8844	M	61.0	4	Single	Lebanon	Human Resour...	Beh
22	0.189318	EID_1536	F	60.0	4	Married	Springfield	IT	Con
23	0.216278	EID_8822	M	60.0	1	Single	Washington	Operarions	Dire
24	0.194358	EID_11398	F	42.0	3	Single	Lebanon	Operarions	Dire
25	0.178116	EID_12965	F	32.0	4	Married	Washington	Operarions	Con
26	0.148686	EID_5204	M	47.0	4	Single	Franklin	Accounting an...	Ana



RESULT

Thus, the employee attrition prediction using orange data mining was executed successfully.

EXP. NO: 13 **DATE:**

VIRAL POST PREDICTION USING ORANGE DATA MINING

AIM

To implement the viral post prediction using orange data mining.

PROCEDURE

Step 1: Upload Dataset

- Click on File. This will put the widget File on the canvas.
- Double click File to open its properties tab.
- Click on the button to navigate to your csv files.
- Connect the widget to the file [Dataset]_Module8_Train(Employee).csv

Step 2: Exclude Column

- The columns Popular and shares are both labels.
- Since they are both labels, we need to drop the column ‘shares’.
- Use select column widget to exclude columns.

Step 3: Edit Metadata

- Change the Feature Type for Popular, from Numeric Feature to Categorical Label.
- Since the column popular is a label.

Step 4: Get Top Features

- Consider only those few features that are very important.
- The important features will greatly improve the performance of the model.
- Those that do not greatly improve our models will be excluded, because we can make our computations faster with fewer features.
- We can do that easily by using Rank Widget.

Step 5: Split Data

- Split the TrainingData_Attrition.csv into both the training data and validation data.
- To do this, we can use Data Sampler.
- Connect widget Select Columns to widget Data Sampler.
- Set the ratio for training data and testing data. Normally it is 70:30 or 80:20.

Step 6: Train Model

- Train the model to find its own rules.
- Use the widget called Test and Score.
- For training purposes, we need to have 2 inputs, one for the training data and another for the learning algorithm.
- We can insert more than 1 learning algorithm if we want to do a comparison among them.

- For the training data, we will use the Data Sample from Data Sampler, which represents the 80% data.
- Connect widget Data Sampler to widget Test and Score.

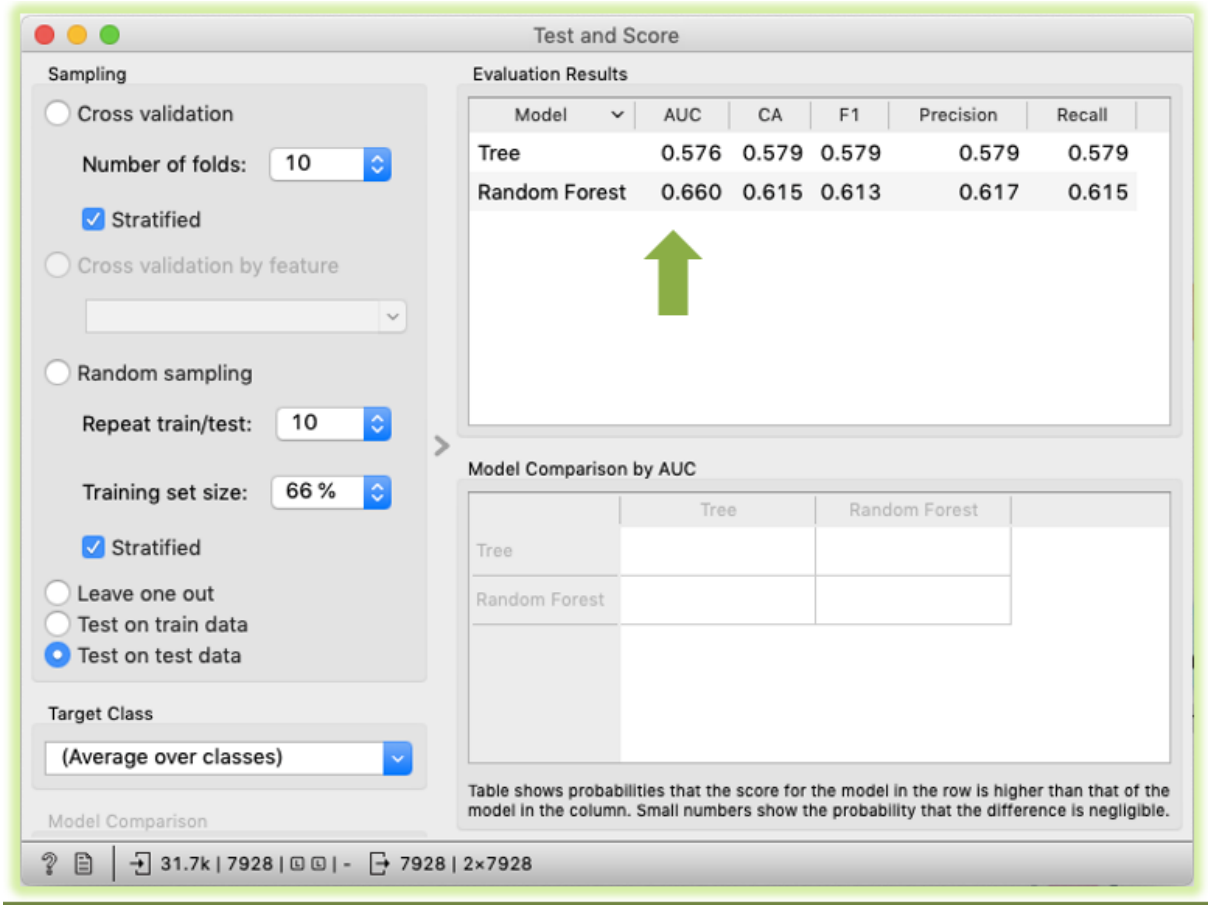
Step 7: Score Model

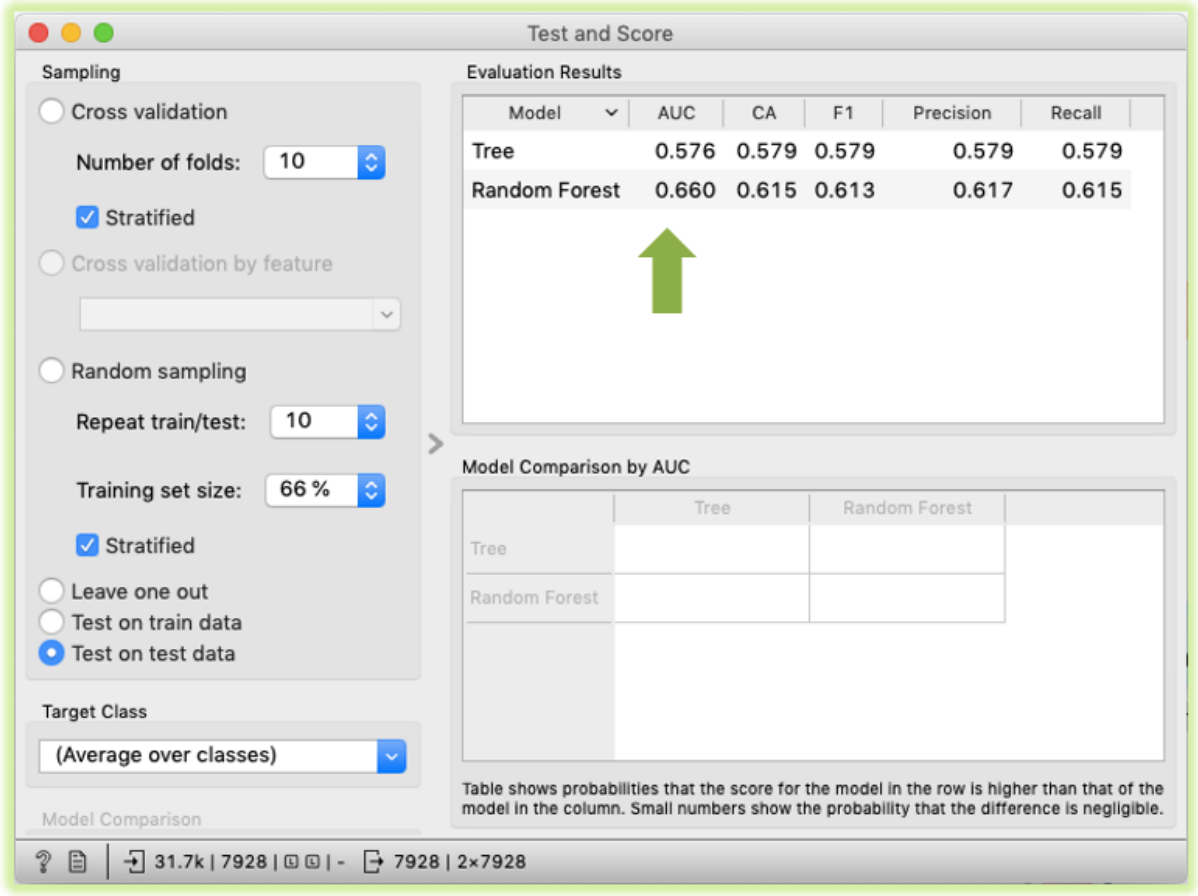
- To verify the performance of the model this is used.
- Since we will be using this to verify the performance, we will connect it to the Remaining Data of Data Sampler.

Step 8: Predictions

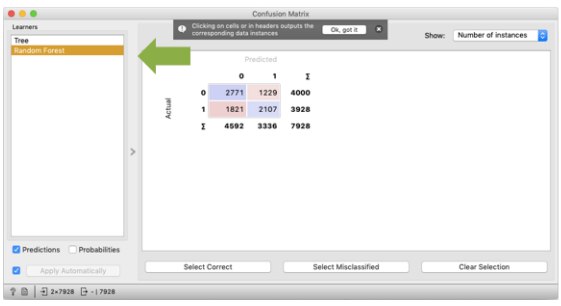
- Use the trained model to make predictions.
- Testing data will be the input to the prediction widget.

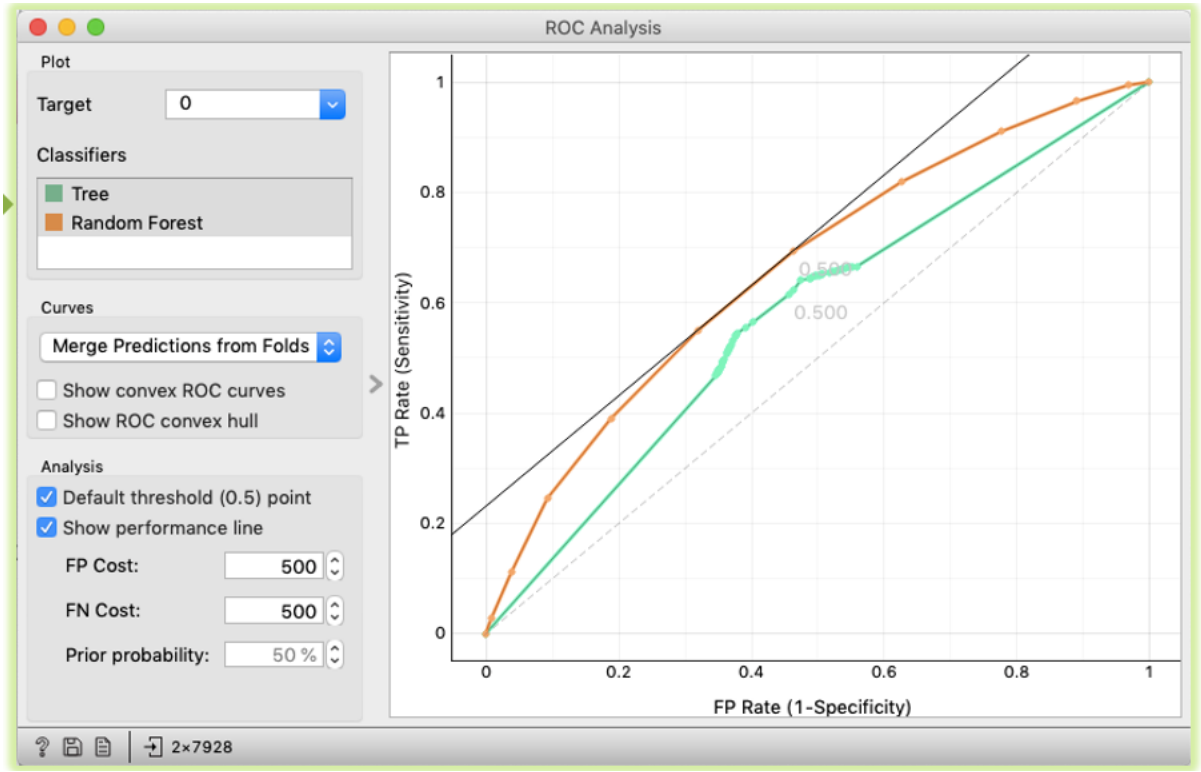
OUTPUT




Test and Score
Sampling
Cross validation
Number of folds: 10
Stratified
Cross validation by feature
Random sampling
Repeat train/test: 10
Training set size: 66 %
Stratified
Leave one out
Test on train data
Test on test data
Target Class
(Average over classes)
Model Comparison
Evaluation Results
Model AUC CA F1 Precision Recall
Tree 0.576 0.579 0.579 0.579 0.579
Random Forest 0.660 0.615 0.613 0.617 0.615
Model Comparison by AUC
Tree Random Forest
Tree
Random Forest
Table shows probabilities that the score for the model in the row is higher than that of the model in the column. Small numbers show the probability that the difference is negligible.
31.7k | 7928 | 7928 | 2×7928


Confusion Matrix
Learners
Tree
Random Forest
Show: Number of instances
Predicted
Actual
0 1 Σ
0 2457 1543 4000
1 1794 2134 3928
Σ 4251 3677 7928
Predictions
Probabilities
Apply Automatically
Select Correct
Select Misclassified
Clear Selection


Confusion Matrix
Learners
Tree
Random Forest
Show: Number of instances
Predicted
Actual
0 1 Σ
0 2771 1229 4000
1 1821 2107 3928
Σ 4592 3336 7928
Predictions
Probabilities
Apply Automatically
Select Correct
Select Misclassified
Clear Selection



RESULT

Thus, the Viral post prediction using orange data mining was executed successfully.