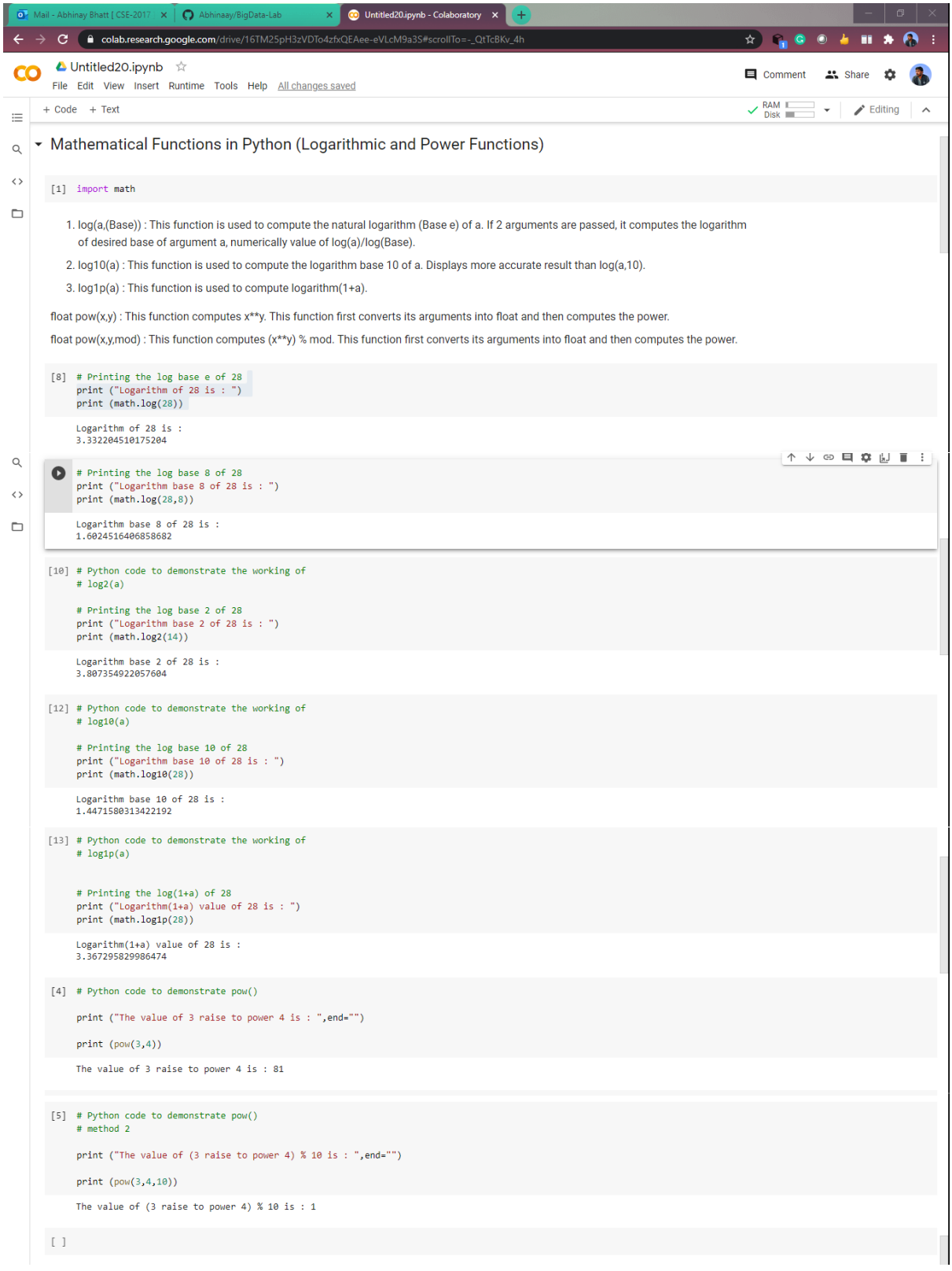

Mail - Abhinay Bhatt [CSE-2017
Abhinaay/BigData-Lab
Untitled20.ipynb - Colaboratory
colab.research.google.com/drive/16TM25pH3zVDTo4zfxQEAee-eVLcM9a3S#scrollTo=-_QtTcBKv_4h
Untitled20.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk
Editing
Mathematical Functions in Python (Logarithmic and Power Functions)
[1] import math
1. log(a,(Base)) : This function is used to compute the natural logarithm (Base e) of a. If 2 arguments are passed, it computes the logarithm of desired base of argument a, numerically value of log(a)/log(Base).
2. log10(a) : This function is used to compute the logarithm base 10 of a. Displays more accurate result than log(a,10).
3. log1p(a) : This function is used to compute logarithm(1+a).
float pow(x,y) : This function computes x**y. This function first converts its arguments into float and then computes the power.
float pow(x,y,mod) : This function computes (x**y) % mod. This function first converts its arguments into float and then computes the power.
[8] # Printing the log base e of 28
print ("Logarithm of 28 is : ")
print (math.log(28))
Logarithm of 28 is :
3.332204510175204
Printing the log base 8 of 28
print ("Logarithm base 8 of 28 is : ")
print (math.log(28,8))
Logarithm base 8 of 28 is :
1.6024516406858682
[10] # Python code to demonstrate the working of
log2(a)
Printing the log base 2 of 28
print ("Logarithm base 2 of 28 is : ")
print (math.log2(14))
Logarithm base 2 of 28 is :
3.807354922057604
[12] # Python code to demonstrate the working of
log10(a)
Printing the log base 10 of 28
print ("Logarithm base 10 of 28 is : ")
print (math.log10(28))
Logarithm base 10 of 28 is :
1.4471580313422192
[13] # Python code to demonstrate the working of
log1p(a)
Printing the log(1+a) of 28
print ("Logarithm(1+a) value of 28 is : ")
print (math.log1p(28))
Logarithm(1+a) value of 28 is :
3.367295829986474
[4] # Python code to demonstrate pow()
print ("The value of 3 raise to power 4 is : ",end="")
print (pow(3,4))
The value of 3 raise to power 4 is : 81
[5] # Python code to demonstrate pow()
method 2
print ("The value of (3 raise to power 4) % 10 is : ",end="")
print (pow(3,4,10))
The value of (3 raise to power 4) % 10 is : 1
[ ]

Untitled21.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share Connect Editing

+ Code + Text

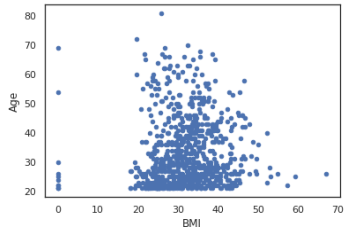
Data Visualization using **Diabetes** dataset

```
# First, we'll import pandas, a data processing and CSV file I/O library
import pandas as pd

# We'll also import seaborn, a Python graphing library
import warnings # current version of seaborn generates a bunch of warnings that we'll ignore
warnings.filterwarnings("ignore")
import seaborn as sns
import matplotlib.pyplot as plt
sns.set(style="white", color_codes=True)

# Next, we'll load the diabetes dataset
diab = pd.read_csv("diabetes.csv") # the iris dataset is now a Pandas DataFrame

diab.head()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

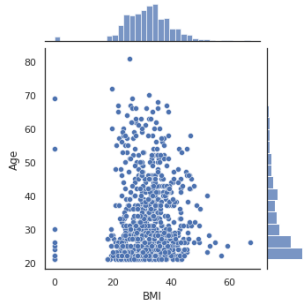
```
# The first way we can plot things is using the .plot extension from Pandas dataframes
# We'll use this to make a scatterplot of the diabetes features.
diab.plot(kind="scatter", x="BMI", y="Age")
```

```
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping will have precedence in case its length matches with *x* &
<matplotlib.axes._subplots.AxesSubplot at 0x7f8f1ab13be0>
```



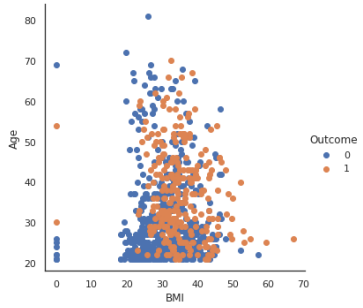

```
[ ] # We can also use the seaborn library to make a similar plot
    # A seaborn jointplot shows bivariate scatterplots and univariate histograms in the same figure
    sns.jointplot(x="BMI", y="Age", data=diab, size=5)
```

```
<seaborn.axisgrid.JointGrid at 0x7f8f170b3c18>
```



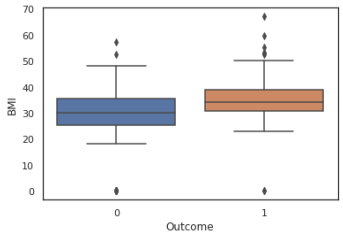

```
[ ] # We'll use seaborn's FacetGrid to color the scatterplot by species
    sns.FacetGrid(diab, hue="Outcome", size=5) \
       .map(plt.scatter, "BMI", "Age") \
       .add_legend()
```

```
<seaborn.axisgrid.FacetGrid at 0x7f8f16f1d4e0>
```

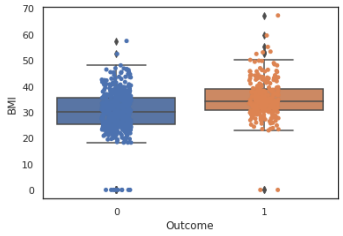


```
[ ] # We can look at an individual feature in Seaborn through a boxplot
    sns.boxplot(x="Outcome", y="BMI", data=diab)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8f16f5b940>
```

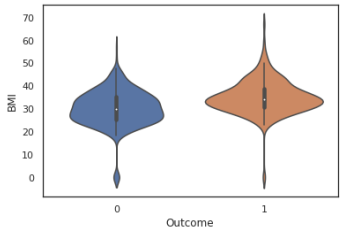



```
[ ] # One way we can extend this plot is adding a layer of individual points on top of
    # it through Seaborn's striplot
    #
    # We'll use jitter=True so that all the points don't fall in single vertical lines
    # above the species
    #
    # Saving the resulting axes as ax each time causes the resulting plot to be shown
    # on top of the previous axes
    ax = sns.boxplot(x="Outcome", y="BMI", data=diab)
    ax = sns.stripplot(x="Outcome", y="BMI", data=diab, jitter=True, edgecolor="gray")
```



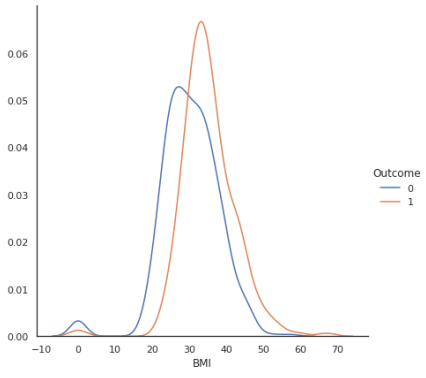

```
[ ] # A violin plot combines the benefits of the previous two plots and simplifies them
    # Denser regions of the data are fatter, and sparser thiner in a violin plot
    sns.violinplot(x="Outcome", y="BMI", data=diab, size=6)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8f16e18240>
```



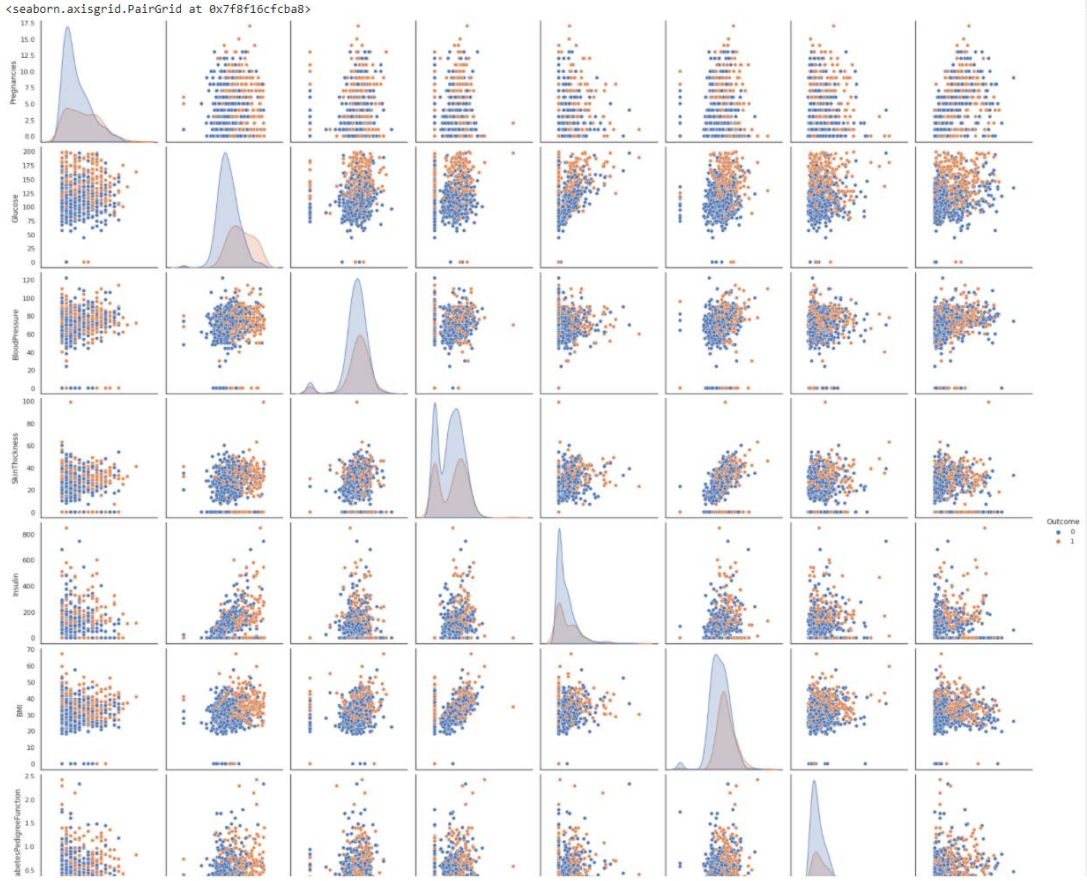

```
[ ] # A final seaborn plot useful for looking at univariate relations is the kdeplot,
    # which creates and visualizes a kernel density estimate of the underlying feature
    sns.FacetGrid(diab, hue="Outcome", size=6) \
       .map(sns.kdeplot, "BMI") \
       .add_legend()
```

```
<seaborn.axisgrid.FacetGrid at 0x7f8f16e23d68>
```



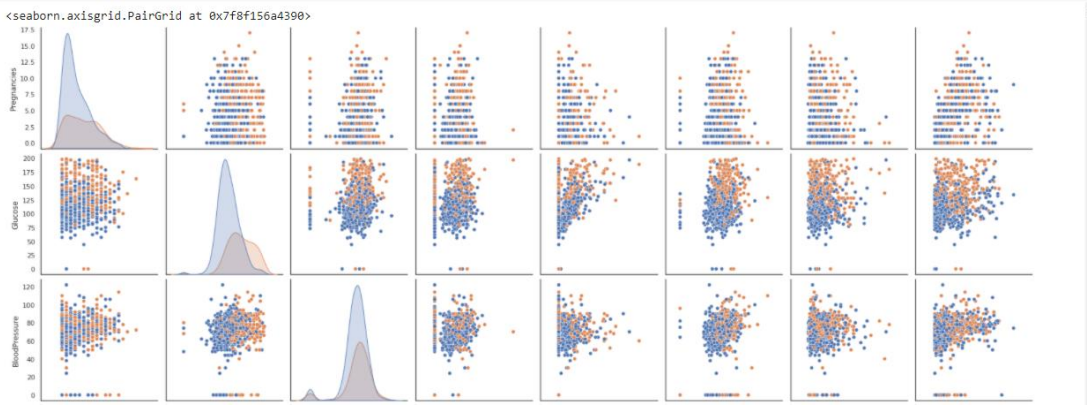
```
[ ]  # Another useful seaborn plot is the pairplot, which shows the bivariate relation
     # between each pair of features
     #
     sns.pairplot(diab, hue="Outcome", size=3)
```

```
<seaborn.axisgrid.PairGrid at 0x7f8f16cfcba8>
```

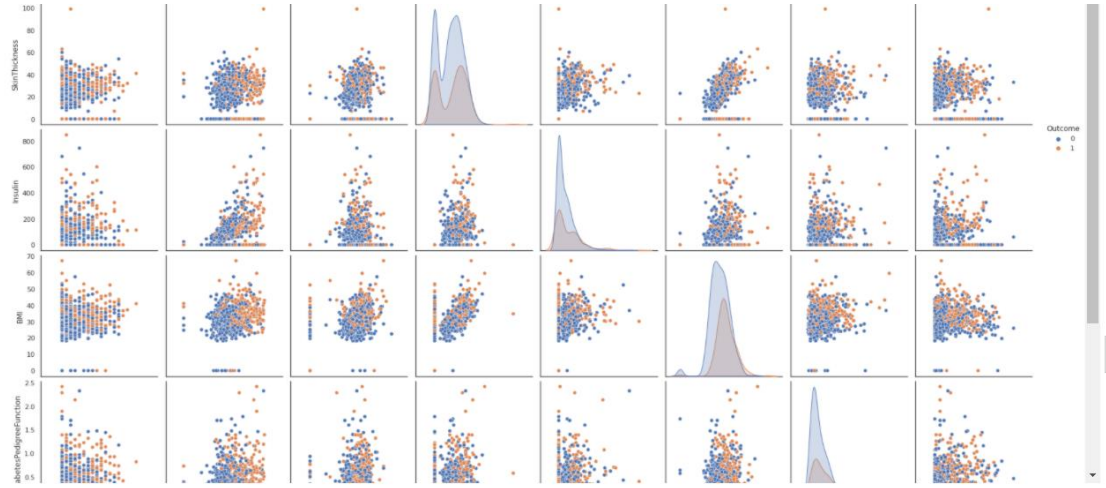



```
[ ]  # The diagonal elements in a pairplot show the histogram by default
     # We can update these elements to show other things, such as a kde
     sns.pairplot(diab, hue="Outcome", size=3, diag_kind="kde")
```

```
<seaborn.axisgrid.PairGrid at 0x7f8f156a4390>
```

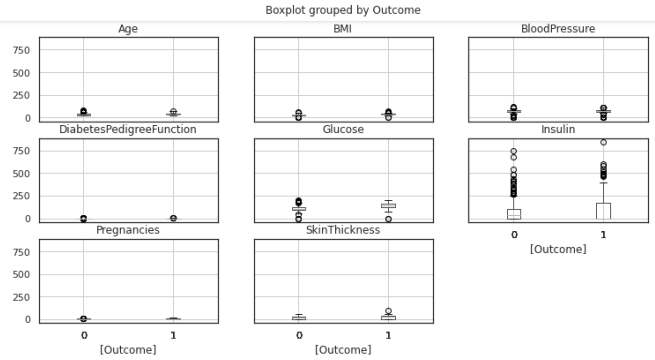


[ ]



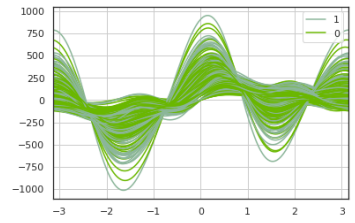
[ ]
```
# Now that we've covered seaborn, let's go back to some of the ones we can make with Pandas
# We can quickly make a boxplot with Pandas on each feature split out by variety
diab.boxplot(by="Outcome", figsize=(12, 6))
```

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7f8f13167780>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f8f12725f28>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f8f126ef5c0>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7f8f1270ba20>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f8f126b8e80>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f8f12676320>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7f8f12624780>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f8f12655ba8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f8f12655c18>]],
      dtype=object)
```



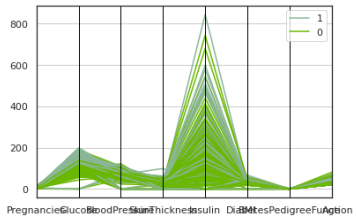
[ ]
```
# One cool more sophisticated technique pandas has available is called Andrews Curves
# Andrews Curves involve using attributes of samples as coefficients for Fourier series
# and then plotting these
from pandas.plotting import andrews_curves
andrews_curves(diab, "Outcome")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8f123259e8>
```



```
[ ] # Another multivariate visualization technique pandas has is parallel_coordinates
    # Parallel coordinates plots each feature on a separate column & then draws lines
    # connecting the features for each data sample
    from pandas.plotting import parallel_coordinates
    parallel_coordinates(diab, "Outcome")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8f11a14710>
```




```
[ ] # A final multivariate visualization technique pandas has is radviz
    # Which puts each feature as a point on a 2D plane, and then simulates
    # having each sample attached to those points through a spring weighted
    # by the relative value for that feature
    from pandas.plotting import radviz
    radviz(diab, "Outcome")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8f110cb9e8>
```


```
[ ]
```

Untitled22.ipynb - Colaboratory

colab.research.google.com/drive/13AR7_uiYvfJ2N9tNVC3cXS3fGOy91FLF#scrollTo=1h10klruKHs6

Untitled22.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

Google Account
Abhinay Bhatt
abhinaybhatt2121999@gmail.com

+ Code + Text

RAM Disk Editing

```
[1] import pandas as pd
    import matplotlib as plt
    from sklearn.tree import DecisionTreeClassifier
    from sklearn.tree import plot_tree, export_graphviz
    from sklearn.datasets import load_iris
    %matplotlib inline
```

```
[2] import numpy as np
    iris = load_iris()
```

```
[3] X,y = load_iris(return_X_y=True)
    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=42)
```

Decision Tree

```
[11] from sklearn.tree import DecisionTreeClassifier
     from sklearn import metrics
```

```
[12] dt=DecisionTreeClassifier()
     dt.fit(X_train, y_train)

     y_pred=dt.predict(X_test)
     print(metrics.accuracy_score(y_test, y_pred)*100)
```


https://accounts.google.com/SignOutOptions?hl=en&continue=https://colab.research.google.com/

Untitled22.ipynb - Colaboratory

colab.research.google.com/drive/13AR7_uiYvfJ2N9tNVC3cXS3fGOy91FLF#scrollTo=1h10klruKHs6

Untitled22.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk Editing

[12] dt=DecisionTreeClassifier()
dt.fit(X_train, y_train)

y_pred=dt.predict(X_test)
print(metrics.accuracy_score(y_test, y_pred)*100)

```
96.0
```

```
from sklearn import tree
tree.plot_tree(dt)
```

```
[Text(181.35000000000002, 201.90857142857143, 'X[3] <= 0.8\ngini = 0.666\nsamples = 100\nvalue = [31, 35, 34]'),
 Text(153.45000000000002, 170.84571428571428, 'gini = 0.0\nsamples = 31\nvalue = [31, 0, 0]'),
 Text(209.25000000000003, 170.84571428571428, 'X[3] <= 1.75\ngini = 0.5\nsamples = 69\nvalue = [0, 35, 34]'),
 Text(139.5, 139.78285714285715, 'X[2] <= 5.35\ngini = 0.188\nsamples = 38\nvalue = [0, 34, 4]'),
 Text(111.60000000000001, 108.72, 'X[0] <= 4.95\ngini = 0.105\nsamples = 36\nvalue = [0, 34, 2]'),
 Text(55.800000000000004, 77.65714285714284, 'X[1] <= 2.45\ngini = 0.5\nsamples = 2\nvalue = [0, 1, 1]'),
 Text(27.900000000000002, 46.59428571428572, 'gini = 0.0\nsamples = 1\nvalue = [0, 1, 0]'),
 Text(83.7, 46.59428571428572, 'gini = 0.0\nsamples = 1\nvalue = [0, 0, 1]'),
 Text(167.4, 77.65714285714284, 'X[1] <= 2.25\ngini = 0.057\nsamples = 34\nvalue = [0, 33, 1]'),
 Text(139.5, 46.59428571428572, 'X[3] <= 1.25\ngini = 0.444\nsamples = 3\nvalue = [0, 2, 1]'),
 Text(111.60000000000001, 15.531428571428563, 'gini = 0.0\nsamples = 2\nvalue = [0, 2, 0]'),
 Text(167.4, 15.531428571428563, 'gini = 0.0\nsamples = 1\nvalue = [0, 0, 1]'),
 Text(195.3, 46.59428571428572, 'gini = 0.0\nsamples = 31\nvalue = [0, 31, 0]'),
 Text(167.4, 108.72, 'gini = 0.0\nsamples = 2\nvalue = [0, 0, 2]'),
 Text(279.0, 139.78285714285715, 'X[2] <= 4.85\ngini = 0.062\nsamples = 31\nvalue = [0, 1, 30]'),
 Text(251.10000000000002, 108.72, 'X[1] <= 3.1\ngini = 0.444\nsamples = 3\nvalue = [0, 1, 2]'),
 Text(223.20000000000002, 77.65714285714284, 'gini = 0.0\nsamples = 2\nvalue = [0, 0, 2]'),
 Text(279.0, 77.65714285714284, 'gini = 0.0\nsamples = 1\nvalue = [0, 1, 0]'),
 Text(306.90000000000003, 108.72, 'gini = 0.0\nsamples = 28\nvalue = [0, 0, 28]')]
```

Activate Windows
Go to Settings to activate Windows.

Untitled22.ipynb - Colaboratory

colab.research.google.com/drive/13AR7_uiYvfJ2N9tNVC3cXS3fGOy91FLF#scrollTo=1h10klruKHs6

Untitled22.ipynb

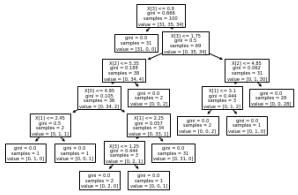
File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk Editing

```
Text(111.60000000000001, 15.531428571428563, 'gini = 0.0\nsamples = 2\nvalue = [0, 2, 0]'),
Text(167.4, 15.531428571428563, 'gini = 0.0\nsamples = 1\nvalue = [0, 0, 1]'),
Text(195.3, 46.59428571428572, 'gini = 0.0\nsamples = 31\nvalue = [0, 31, 0]'),
Text(167.4, 108.72, 'gini = 0.0\nsamples = 2\nvalue = [0, 0, 2]'),
Text(279.0, 139.78285714285715, 'X[2] <= 4.85\ngini = 0.062\nsamples = 31\nvalue = [0, 1, 30]'),
Text(251.10000000000002, 108.72, 'X[1] <= 3.1\ngini = 0.444\nsamples = 3\nvalue = [0, 1, 2]'),
Text(223.20000000000002, 77.65714285714284, 'gini = 0.0\nsamples = 2\nvalue = [0, 0, 2]'),
Text(279.0, 77.65714285714284, 'gini = 0.0\nsamples = 1\nvalue = [0, 1, 0]'),
Text(306.90000000000003, 108.72, 'gini = 0.0\nsamples = 28\nvalue = [0, 0, 28]')]
```



ID3

```
[4] dcf = DecisionTreeClassifier(criterion='entropy')
    dcf.fit(X_train,y_train)
```

```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='entropy',
                       max_depth=None, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2
```


Untitled22.ipynb - Colaboratory

colab.research.google.com/drive/13AR7_uiYvfJ2N9tNVC3cXS3fGOy91FLF#scrollTo=ilxBTD5FI9TV

Untitled22.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk Editing

ID3

[4] dcf = DecisionTreeClassifier(criterion='entropy')
dcf.fit(X_train,y_train)

```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='entropy',
                       max_depth=None, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
```

```
[15] pred = dcf.predict(X_test)
```

```
[17] from sklearn.metrics import classification_report
```

```
[18] print(classification_report(y_true=y_test, y_pred=pred, target_names=iris.target_names))
```

```
                precision    recall  f1-score   support

        setosa       1.00      1.00      1.00        19
    versicolor       0.94      1.00      0.97        15
     virginica       1.00      0.94      0.97        16

      accuracy                           0.98        50
     macro avg       0.98      0.98      0.98        50
  weighted avg       0.98      0.98      0.98        50
```

Untitled22.ipynb - Colaboratory

colab.research.google.com/drive/13AR7_uiYvfJ2N9tNVC3cXS3fGOy91FLF#scrollTo=ilxBTD5FI9TV

Untitled22.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk Editing

```
[17] from sklearn.metrics import classification_report
```

```
[18] print(classification_report(y_true=y_test, y_pred=pred, target_names=iris.target_names))
```

```
                  precision    recall  f1-score   support

          setosa       1.00      1.00      1.00        19
      versicolor       0.94      1.00      0.97        15
       virginica       1.00      0.94      0.97        16

        accuracy                           0.98        50
       macro avg       0.98      0.98      0.98        50
    weighted avg       0.98      0.98      0.98        50
```

```
[6] from sklearn import tree
```

```
tree.plot_tree(dcf)
```

```
[Text(148.8, 201.90857142857143, 'X[2] <= 2.45\nentropy = 1.583\nsamples = 100\nvalue = [31, 35, 34]'),
 Text(111.60000000000001, 170.84571428571428, 'entropy = 0.0\nsamples = 31\nvalue = [31, 0, 0]'),
 Text(186.0, 170.84571428571428, 'X[3] <= 1.75\nentropy = 1.0\nsamples = 69\nvalue = [0, 35, 34]'),
 Text(111.60000000000001, 139.78285714285715, 'X[2] <= 5.35\nentropy = 0.485\nsamples = 38\nvalue = [0, 34, 4]'),
 Text(74.4, 108.72, 'X[3] <= 1.45\nentropy = 0.31\nsamples = 36\nvalue = [0, 34, 2]'),
 Text(37.2, 77.65714285714284, 'entropy = 0.0\nsamples = 26\nvalue = [0, 26, 0]'),
 Text(111.60000000000001, 77.65714285714284, 'X[1] <= 2.6\nentropy = 0.722\nsamples = 10\nvalue = [0, 8, 2]'),
 Text(74.4, 46.59428571428572, 'X[0] <= 6.15\nentropy = 0.918\nsamples = 3\nvalue = [0, 1, 2]'),
 Text(37.2, 15.531428571428563, 'entropy = 0.0\nsamples = 2\nvalue = [0, 0, 2]'),
 Text(111.60000000000001, 15.531428571428563, 'entropy = 0.0\nsamples = 1\nvalue = [0, 1, 0]'),
 Text(148.8, 46.59428571428572, 'entropy = 0.0\nsamples = 7\nvalue = [0, 7, 0]'),
```


Untitled22.ipynb - Colaboratory

colab.research.google.com/drive/13AR7_uiYvfJ2N9tNVC3cXS3fGOy91FLF#scrollTo=ilxBTD5FI9TV

Untitled22.ipynb

File Edit View Insert Runtime Tools Help All changes saved

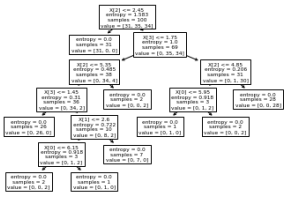
Comment Share

+ Code + Text

RAM Disk Editing

tree.plot_tree(dcf)

```
[Text(148.8, 201.90857142857143, 'X[2] <= 2.45\nentropy = 1.583\nsamples = 100\nvalue = [31, 35, 34]'),
 Text(111.60000000000001, 170.84571428571428, 'entropy = 0.0\nsamples = 31\nvalue = [31, 0, 0]'),
 Text(186.0, 170.84571428571428, 'X[3] <= 1.75\nentropy = 1.0\nsamples = 69\nvalue = [0, 35, 34]'),
 Text(111.60000000000001, 139.78285714285715, 'X[2] <= 5.35\nentropy = 0.485\nsamples = 38\nvalue = [0, 34, 4]'),
 Text(74.4, 108.72, 'X[3] <= 1.45\nentropy = 0.31\nsamples = 36\nvalue = [0, 34, 2]'),
 Text(37.2, 77.65714285714284, 'entropy = 0.0\nsamples = 26\nvalue = [0, 26, 0]'),
 Text(111.60000000000001, 77.65714285714284, 'X[1] <= 2.6\nentropy = 0.722\nsamples = 10\nvalue = [0, 8, 2]'),
 Text(74.4, 46.59428571428572, 'X[0] <= 6.15\nentropy = 0.918\nsamples = 3\nvalue = [0, 1, 2]'),
 Text(37.2, 15.531428571428563, 'entropy = 0.0\nsamples = 2\nvalue = [0, 0, 2]'),
 Text(111.60000000000001, 15.531428571428563, 'entropy = 0.0\nsamples = 1\nvalue = [0, 1, 0]'),
 Text(148.8, 46.59428571428572, 'entropy = 0.0\nsamples = 7\nvalue = [0, 7, 0]'),
 Text(148.8, 108.72, 'entropy = 0.0\nsamples = 2\nvalue = [0, 0, 2]'),
 Text(260.40000000000003, 139.78285714285715, 'X[2] <= 4.85\nentropy = 0.206\nsamples = 31\nvalue = [0, 1, 30]'),
 Text(223.20000000000002, 108.72, 'X[0] <= 5.95\nentropy = 0.918\nsamples = 3\nvalue = [0, 1, 2]'),
 Text(186.0, 77.65714285714284, 'entropy = 0.0\nsamples = 1\nvalue = [0, 1, 0]'),
 Text(260.40000000000003, 77.65714285714284, 'entropy = 0.0\nsamples = 2\nvalue = [0, 0, 2]'),
 Text(297.6, 108.72, 'entropy = 0.0\nsamples = 28\nvalue = [0, 0, 28]')]
```



Untitled22.ipynb - Colaboratory

colab.research.google.com/drive/13AR7_uiYvfJ2N9tNVC3cXS3fGOy91FLF#scrollTo=ilxBTD5FI9TV

Untitled22.ipynb

File Edit View Insert Runtime Tools Help Saving...

Comment Share

+ Code + Text

RAM Disk Editing

GINI Index

```
[14] dcf = DecisionTreeClassifier(criterion='gini')
     dcf.fit(X_train,y_train)
```

```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=None, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
```

```
[20] print(classification_report(y_true=y_test, y_pred=pred, target_names=iris.target_names))
```

```
                precision    recall  f1-score   support

        setosa       1.00      1.00      1.00        19
    versicolor       0.94      1.00      0.97        15
     virginica       1.00      0.94      0.97        16

      accuracy                           0.98        50
     macro avg       0.98      0.98      0.98        50
  weighted avg       0.98      0.98      0.98        50
```

```
tree.plot_tree(dcf)
```

```
[Text(181.35000000000002, 201.90857142857143, 'X[2] <= 2.45\ngini = 0.666\nsamples = 100\nvalue = [31, 35, 34]'),
 Text(153.45000000000002, 170.84571428571428, 'gini = 0.0\nsamples = 31\nvalue = [31, 0, 0]'),
 Text(209.25000000000003, 170.84571428571428, 'X[3] <= 1.75\ngini = 0.5\nsamples = 69\nvalue = [0, 35, 34]'),
```


Untitled22.ipynb - Colaboratory

colab.research.google.com/drive/13AR7_uiYvfJ2N9tNVC3cXS3fGOy91FLF#scrollTo=ilxBTD5FI9TV

Untitled22.ipynb

File Edit View Insert Runtime Tools Help All changes saved

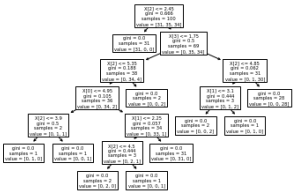
Comment Share

+ Code + Text

RAM Disk Editing

```
[21] tree.plot_tree(dcf)
```

```
[Text(181.35000000000002, 201.90857142857143, 'X[2] <= 2.45\ngini = 0.666\nsamples = 100\nvalue = [31, 35, 34]'),
 Text(153.45000000000002, 170.84571428571428, 'gini = 0.0\nsamples = 31\nvalue = [31, 0, 0]'),
 Text(209.25000000000003, 170.84571428571428, 'X[3] <= 1.75\ngini = 0.5\nsamples = 69\nvalue = [0, 35, 34]'),
 Text(139.5, 139.78285714285715, 'X[2] <= 5.35\ngini = 0.188\nsamples = 38\nvalue = [0, 34, 4]'),
 Text(111.60000000000001, 108.72, 'X[0] <= 4.95\ngini = 0.105\nsamples = 36\nvalue = [0, 34, 2]'),
 Text(55.800000000000004, 77.65714285714284, 'X[2] <= 3.9\ngini = 0.5\nsamples = 2\nvalue = [0, 1, 1]'),
 Text(27.900000000000002, 46.59428571428572, 'gini = 0.0\nsamples = 1\nvalue = [0, 1, 0]'),
 Text(83.7, 46.59428571428572, 'gini = 0.0\nsamples = 1\nvalue = [0, 0, 1]'),
 Text(167.4, 77.65714285714284, 'X[1] <= 2.25\ngini = 0.057\nsamples = 34\nvalue = [0, 33, 1]'),
 Text(139.5, 46.59428571428572, 'X[2] <= 4.5\ngini = 0.444\nsamples = 3\nvalue = [0, 2, 1]'),
 Text(111.60000000000001, 15.531428571428563, 'gini = 0.0\nsamples = 2\nvalue = [0, 2, 0]'),
 Text(167.4, 15.531428571428563, 'gini = 0.0\nsamples = 1\nvalue = [0, 0, 1]'),
 Text(195.3, 46.59428571428572, 'gini = 0.0\nsamples = 31\nvalue = [0, 31, 0]'),
 Text(167.4, 108.72, 'gini = 0.0\nsamples = 2\nvalue = [0, 0, 2]'),
 Text(279.0, 139.78285714285715, 'X[2] <= 4.85\ngini = 0.062\nsamples = 31\nvalue = [0, 1, 30]'),
 Text(251.10000000000002, 108.72, 'X[1] <= 3.1\ngini = 0.444\nsamples = 3\nvalue = [0, 1, 2]'),
 Text(223.20000000000002, 77.65714285714284, 'gini = 0.0\nsamples = 2\nvalue = [0, 0, 2]'),
 Text(279.0, 77.65714285714284, 'gini = 0.0\nsamples = 1\nvalue = [0, 1, 0]'),
 Text(306.90000000000003, 108.72, 'gini = 0.0\nsamples = 28\nvalue = [0, 0, 28]')]
```



Untitled22.ipynb - Colaboratory

colab.research.google.com/drive/13AR7_uiYvfJ2N9tNVC3cXS3fGOy91FLF#scrollTo=xzjr-JrmJ_UD

Untitled22.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk Editing

CART

```
[23] from sklearn.tree import DecisionTreeRegressor
     rng=np.random.RandomState(6)
     X=np.sort(5*rng.rand(80,1), axis=0)
     y=np.sin(X).ravel()
```

```
[24] cart=tree.DecisionTreeRegressor(max_depth=4)
```

```
[25] cart.fit(X_train, y_train)

     DecisionTreeRegressor(ccp_alpha=0.0, criterion='mse', max_depth=4,
                           max_features=None, max_leaf_nodes=None,
                           min_impurity_decrease=0.0, min_impurity_split=None,
                           min_samples_leaf=1, min_samples_split=2,
                           min_weight_fraction_leaf=0.0, presort='deprecated',
                           random_state=None, splitter='best')
```

```
[26] pred=cart.predict(X_test)
```

```
pred

array([1.02941176, 0.        , 2.        , 1.02941176, 1.02941176,
       0.        , 1.02941176, 2.        , 1.02941176, 1.02941176,
       2.        , 0.        , 0.        , 0.        , 0.        ,
       1.02941176, 2.        , 1.02941176, 1.02941176, 2.        ,
       0.        , 2.        , 0.        , 2.        , 2.        ,
       2.        , 2.        , 2.        , 0.        , 0.        ,
       0.        , 0.        , 1.02941176, 0.        , 0.        ,
```


Untitled22.ipynb - Colaboratory

colab.research.google.com/drive/13AR7_uiYvfJ2N9tNVC3cXS3fGOy91FLF#scrollTo=xzjr-JrmJ_UD

Untitled22.ipynb

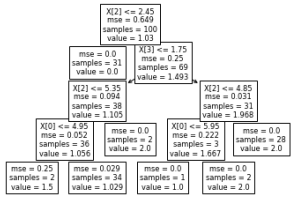
File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk Editing

```
[28] tree.plot_tree(cart)

     [Text(148.8, 195.696, 'X[2] <= 2.45\nmse = 0.649\nsamples = 100\nvalue = 1.03'),
      Text(111.60000000000001, 152.208, 'mse = 0.0\nsamples = 31\nvalue = 0.0'),
      Text(186.0, 152.208, 'X[3] <= 1.75\nmse = 0.25\nsamples = 69\nvalue = 1.493'),
      Text(111.60000000000001, 108.72, 'X[2] <= 5.35\nmse = 0.094\nsamples = 38\nvalue = 1.105'),
      Text(74.4, 65.232, 'X[0] <= 4.95\nmse = 0.052\nsamples = 36\nvalue = 1.056'),
      Text(37.2, 21.744, 'mse = 0.25\nsamples = 2\nvalue = 1.5'),
      Text(111.60000000000001, 21.744, 'mse = 0.029\nsamples = 34\nvalue = 1.029'),
      Text(148.8, 65.232, 'mse = 0.0\nsamples = 2\nvalue = 2.0'),
      Text(260.40000000000003, 108.72, 'X[2] <= 4.85\nmse = 0.031\nsamples = 31\nvalue = 1.968'),
      Text(223.20000000000002, 65.232, 'X[0] <= 5.95\nmse = 0.222\nsamples = 3\nvalue = 1.667'),
      Text(186.0, 21.744, 'mse = 0.0\nsamples = 1\nvalue = 1.0'),
      Text(260.40000000000003, 21.744, 'mse = 0.0\nsamples = 2\nvalue = 2.0'),
      Text(297.6, 65.232, 'mse = 0.0\nsamples = 28\nvalue = 2.0')]
```




```
[ ]
```

Abhinay Bhatt - 179301009

Write a program to implement the Naïve Bayesian classifier

```
# Load Dataset
from  sklearn.datasets import load_iris
import pandas as pd
import numpy as np

data = load_iris()
X, y, column_names = data['data'], data['target'], data['feature_names']
X=pd.DataFrame(X, columns = column_names)


# split the data
from sklearn.model_selection import train_test_split
X_train, X_val, y_train, y_val = train_test_split(X,y, random_state = 44)


#basically this is our fit function
# statistics for the train set
means = X_train.groupby(y_train).apply(np.mean)
std=X_train.groupby(y_train).apply(np.std)

# class prior probabilities
probs = X_train.groupby(y_train).apply(lambda x: len(x)) / X_train.shape[0]


from scipy.stats import norm
y_pred = []
# for each element in the validation set
for elem in range(X_val.shape[0]):
  p = {}

  # for each possible class
  for c1 in np.unique(y_train):
    # take the prior probability of the given class
    p[c1] = probs.iloc[c1]

    # for each column in the data
    for index, param in enumerate(X_val.iloc[elem]):
      # multiply by the probability of the given column value to belong to the distributio
      # of the train column for the given class
      p[c1] *= norm.pdf(param, means.iloc[c1, index], std.iloc[c1, index])
  y_pred.append(pd.Series(p).values.argmax())



# my classifier
from sklearn.metrics import accuracy_score
accuracy_score(y_val, y_pred)
```

```
0.9210526315789473
```

```
# sklearn classifier
from sklearn.naive_bayes import GaussianNB
model = GaussianNB()
model.fit(X_train, y_train)

accuracy_score(y_val, model.predict(X_val))
```

```
0.9210526315789473
```

Abhinay Bhatt - 179301009 - CSE A

```
# importing libraries
import pandas as pd
import numpy as np
from sklearn import metrics
import matplotlib.pyplot as plt
import seaborn as sns


from sklearn.datasets import load_boston
boston = load_boston()


data=pd.DataFrame(boston.data)


data.head()
```

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33

```
data.columns=boston.feature_names
data.head()
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	L
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	

Double-click (or enter) to edit

```
# Adding target variable to dataframe
data['PRICE'] = boston.target
# Median value of owner occupied home is $1000
```

```
data.shape
```

```
(506, 14)
```

```
data.columns
```

```
Index(['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD', 'TAX',
       'PTRATIO', 'B', 'LSTAT', 'PRICE'],
      dtype='object')
```

```
data.dtypes
```

```
CRIM       float64
ZN         float64
INDUS      float64
CHAS       float64
NOX        float64
RM         float64
AGE        float64
DIS        float64
RAD        float64
TAX        float64
PTRATIO    float64
B          float64
LSTAT      float64
PRICE      float64
dtype: object
```

```
# identify the unique number of values in the dataset
data.nunique()
```

```
CRIM       504
ZN          26
INDUS       76
CHAS         2
NOX         81
RM         446
AGE        356
DIS        412
RAD          9
TAX         66
PTRATIO     46
B          357
LSTAT      455
PRICE      229
dtype: int64
```

```
# check for missing values
data.isnull().sum()
```

```
CRIM       0
ZN         0
INDUS      0
CHAS       0
NOX        0
```

```
RM          0
AGE         0
DIS         0
RAD         0
TAX         0
PTRATIO     0
B           0
LSTAT       0
PRICE       0
dtype: int64
```

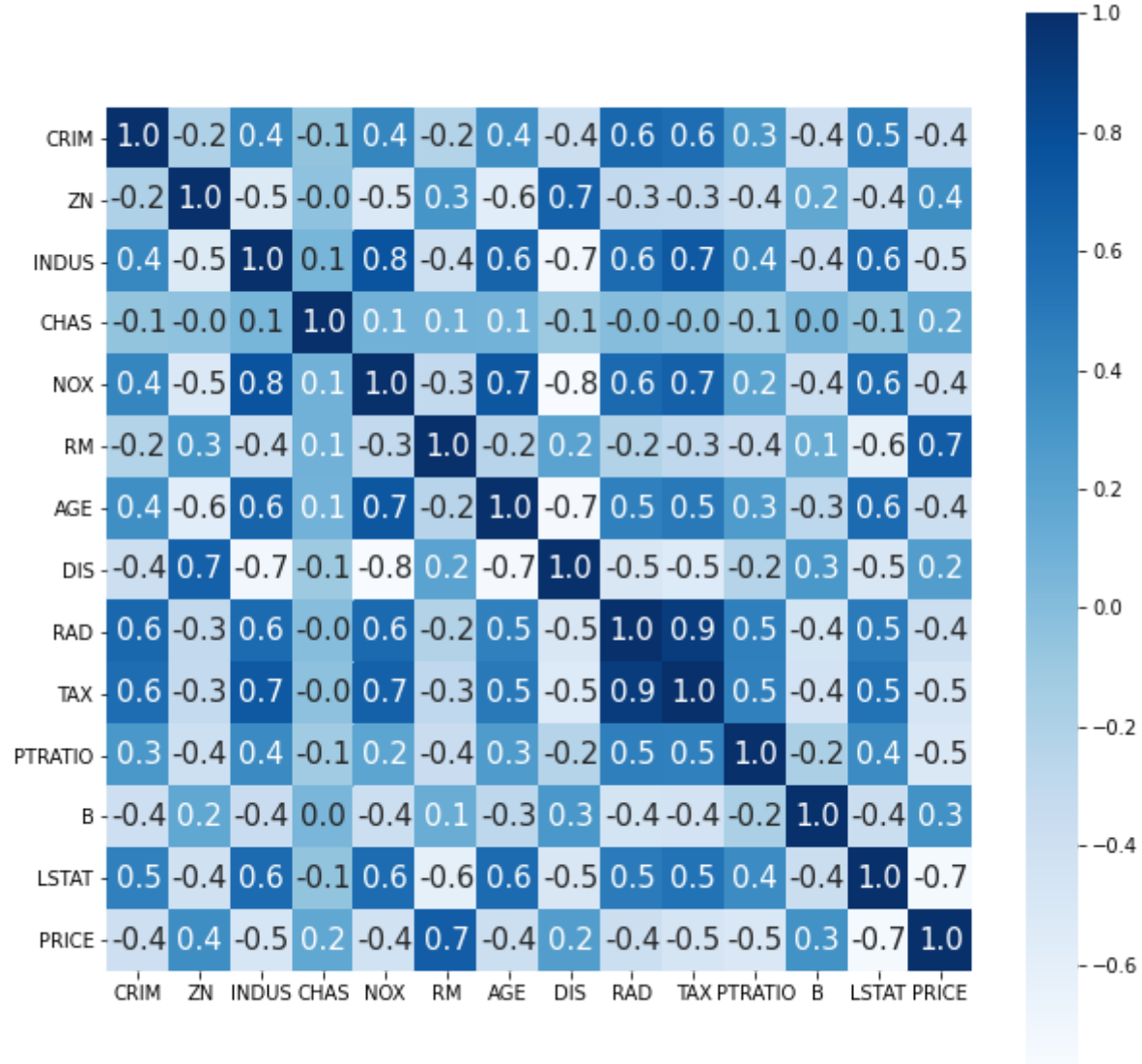
```
# see rows with missing value
data[data.isnull().any(axis=1)]
```

CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	PRICE

```
corr=data.corr()
```

```
plt.figure(figsize=(10,10))
sns.heatmap(corr, cbar=True, square=True, fmt='.1f', annot=True, annot_kws={'size': 15}, c
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f65f170e080>
```



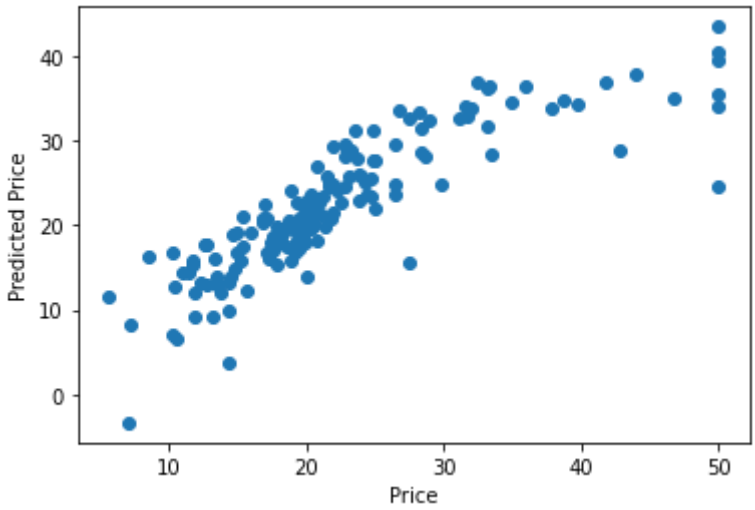
```
X=data.drop(['PRICE'], axis=1)
y=data['PRICE']


# Seperating training and testing data
from sklearn.model_selection import train_test_split


train_data,test_data,train_label,test_label=train_test_split(X,y,test_size=.3)


# calling linear regression model
from sklearn.linear_model import LinearRegression


# model creation
lreg=LinearRegression()


# fitting exp and sal

trained=lreg.fit(train_data,train_label)


pred=trained.predict(test_data)


print(pred)
```

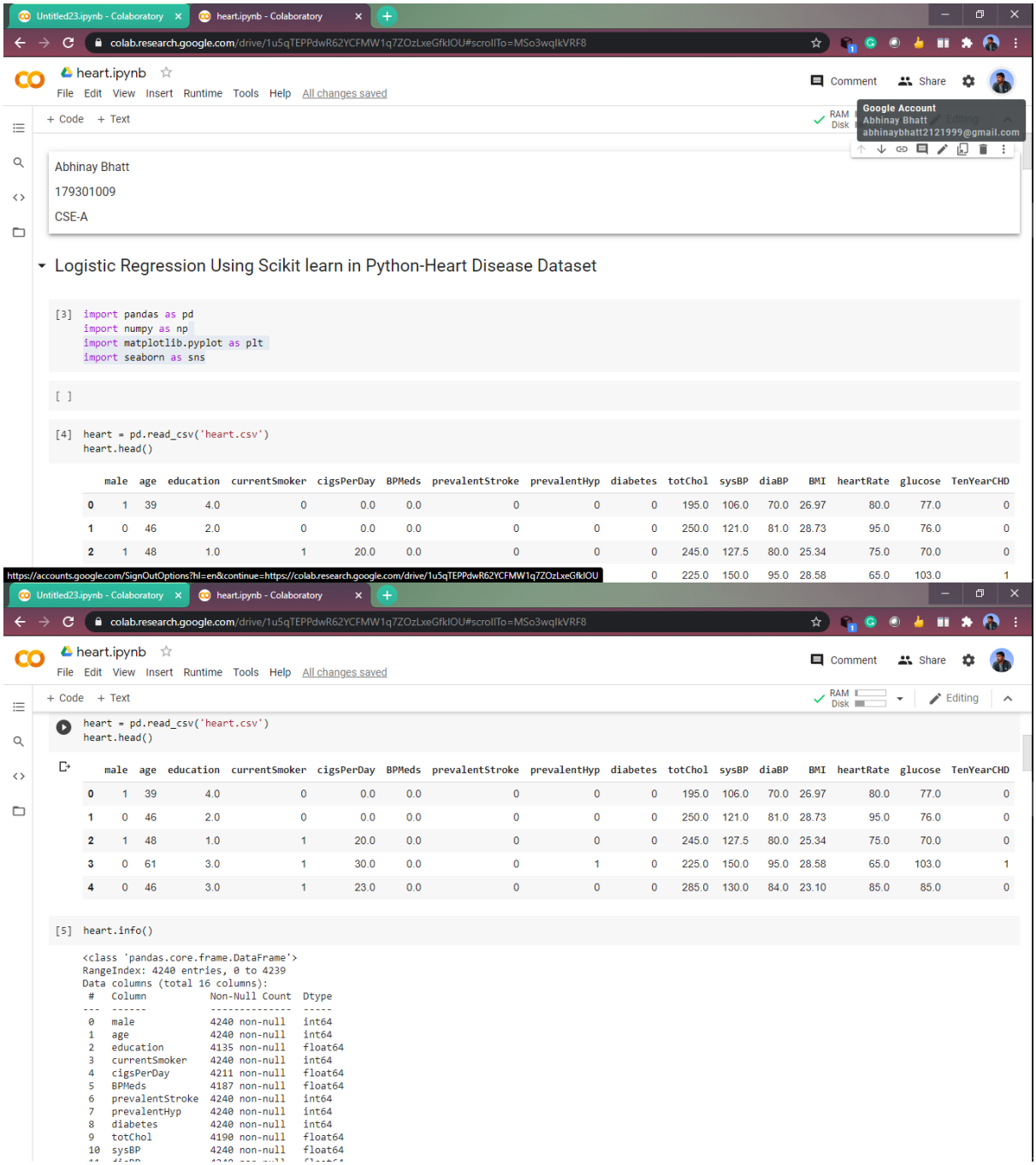
```
     [22.65915902 31.70620455 36.08641044 28.43350977 36.5043496  20.11322198
      35.02084875 23.08248174 21.35460972 19.75553422 19.32391426 25.79228001
      39.40147251 16.75210646 15.55498284 33.22238765 20.71985976 19.20803596
      24.70646436  9.97526812 15.72712897 28.82755667 23.29339815 36.94946097
      15.24878453 19.83431082 13.18856685 14.37884885 25.56852223 13.30839168
      20.69798045 -3.33713438 22.84956782 28.80118428 18.56242196 21.97555848
      28.16387945 32.62525013  9.17036371 19.18975975 25.7802491  15.97708621
      22.67020653 20.94874285 16.74867648 37.84436092 16.85775081 16.0951295
      15.82804713 13.0845633  16.86589227 33.71391128 34.11350584 34.22345212
      19.78132428 15.37643511 25.68130046 23.39352047 22.74273118 12.16242192
      32.65879336 21.05863824 35.44149197 31.24485754 31.21369891 14.01754657
      43.4607242  11.55476502 21.45450906 17.67105095 36.94212756 33.63229374
      34.07959922 17.5111573  32.37073516 14.75800186  3.85805757 31.49138262
      24.5167746  17.21043817 24.60212612  7.03697595 25.88717086 22.9996723
      11.91471648  6.69279979 13.08368567 16.26270669 20.83225491 22.14515528
      17.95944098 36.47634326 25.04974485 28.07681239 20.68806175 22.50664589
       8.34674325 11.95608767 27.9799561  13.16288076 24.90270984 13.90074064
      17.6338948  18.61835629 18.23568497 29.36007359 29.55990884 27.54310523
      19.76614152 17.98326989 19.26321003 19.50865183 22.04288148 24.19054453
      23.60393973 18.82324691 29.44143872 28.6632786  21.15897923 40.35655323
      18.26687685 23.67276741 20.56550688 23.64513462 17.77485525 20.04722661
      28.91443437 27.56398886 15.7446028  12.63282733 34.74047086 17.44727482
      24.47438143 20.40522005 33.71735416 24.74997552 34.50326874 24.21540802
      24.93992385 26.91957858 32.82615266 19.92447919  9.20307924 18.72495925
      18.29477994 16.88952396 14.36109382 22.31296899 19.11258061 16.75413842
      21.20260971 13.83565682]
```

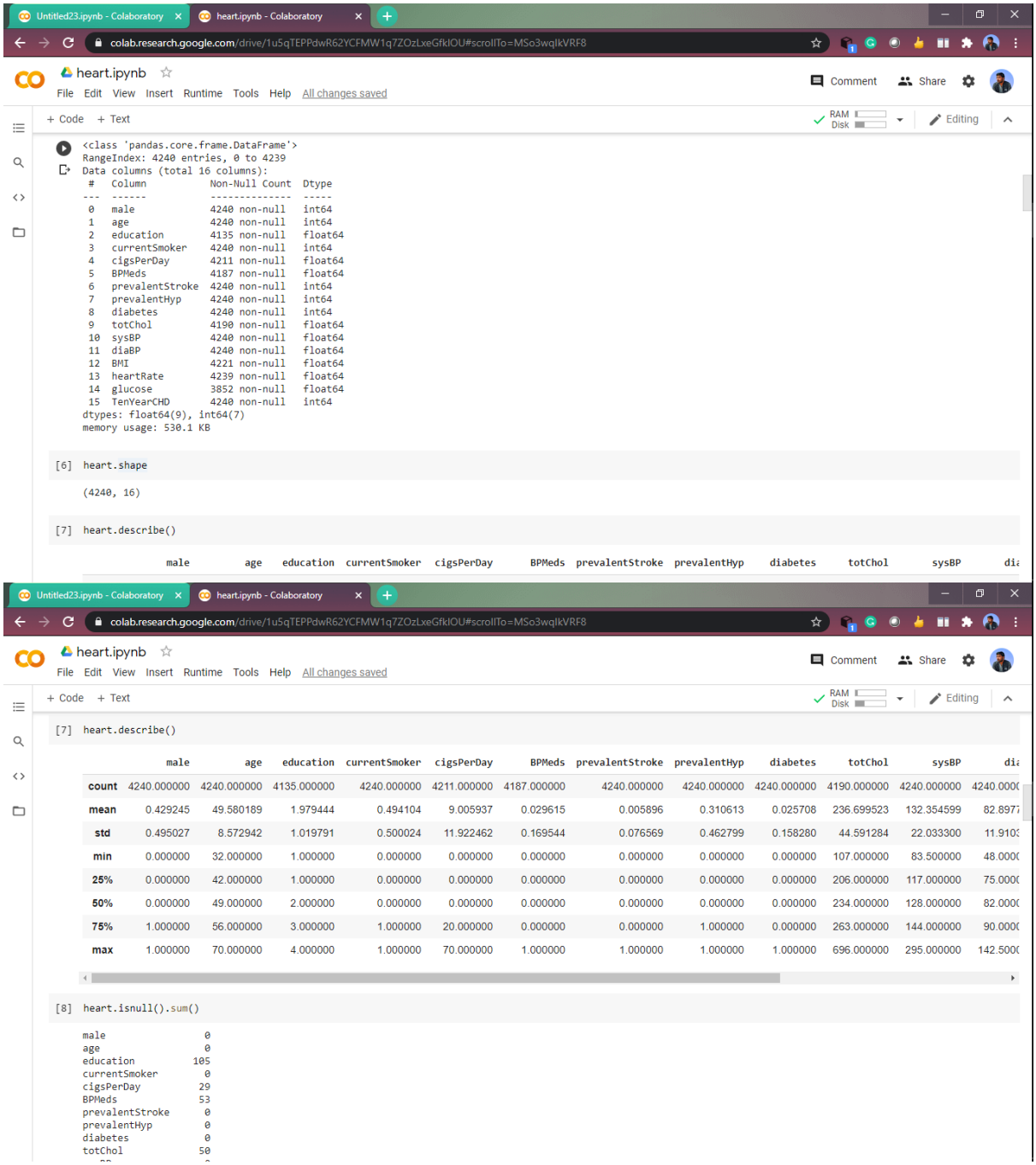
```
plt.scatter(test_label,pred)
```

```
plt.xlabel('Price')
plt.ylabel('Predicted Price')
```

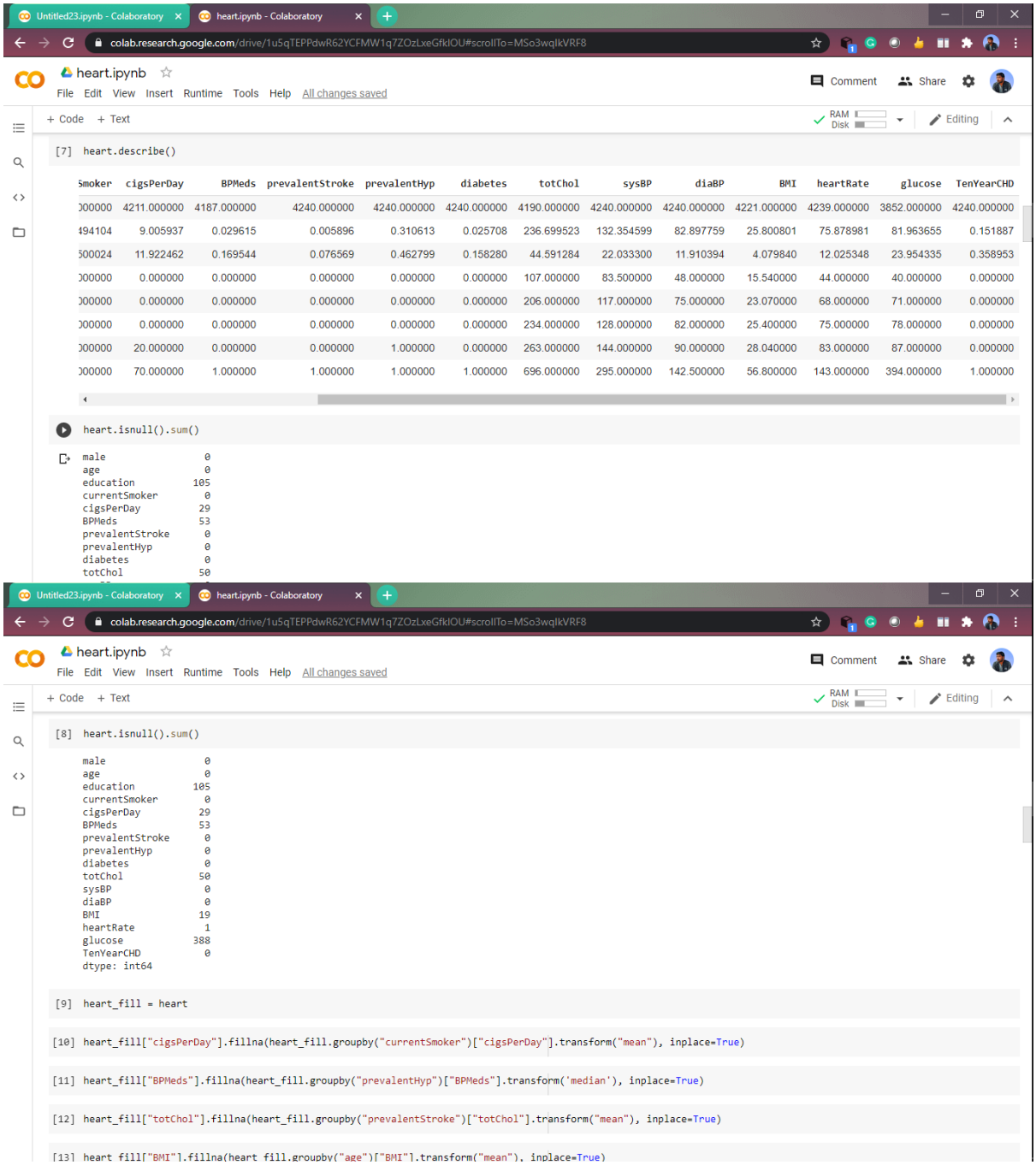
```
Text(0, 0.5, 'Predicted Price')
```



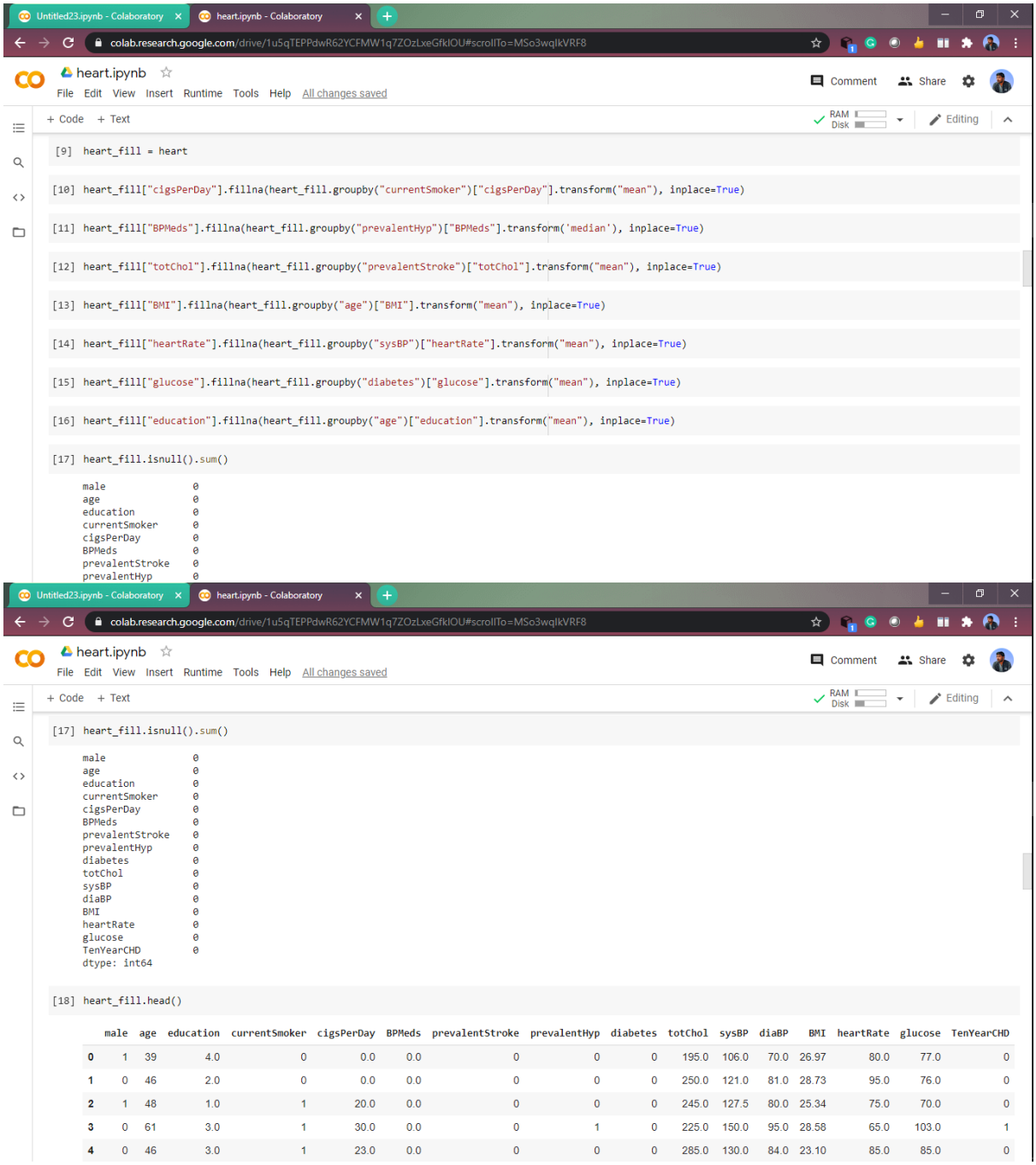

Untitled23.ipynb - Colaboratory
heart.ipynb - Colaboratory
colab.research.google.com/drive/1u5qTEPPdwR62YCFMW1q7ZOzLxeGfkIOU#scrollTo=MSo3wqIkVRF8
heart.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
RAM Disk
Editing
Google Account
Abhinay Bhatt
abhinaybhatt2121999@gmail.com
Abhinay Bhatt
179301009
CSE-A
Logistic Regression Using Scikit learn in Python-Heart Disease Dataset
[3] import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
[ ]
[4] heart = pd.read_csv('heart.csv')
heart.head()
male age education currentSmoker cigsPerDay BPMeds prevalentStroke prevalentHyp diabetes totChol sysBP diaBP BMI heartRate glucose TenYearCHD
0 1 39 4.0 0 0.0 0.0 0 0 0 195.0 106.0 70.0 26.97 80.0 77.0 0
1 0 46 2.0 0 0.0 0.0 0 0 0 250.0 121.0 81.0 28.73 95.0 76.0 0
2 1 48 1.0 1 20.0 0.0 0 0 0 245.0 127.5 80.0 25.34 75.0 70.0 0
https://accounts.google.com/SignOutOptions?hl=en&continue=https://colab.research.google.com/drive/1u5qTEPPdwR62YCFMW1q7ZOzLxeGfkIOU
0 225.0 150.0 95.0 28.58 65.0 103.0 1
Untitled23.ipynb - Colaboratory
heart.ipynb - Colaboratory
colab.research.google.com/drive/1u5qTEPPdwR62YCFMW1q7ZOzLxeGfkIOU#scrollTo=MSo3wqIkVRF8
heart.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
RAM Disk
Editing
heart = pd.read_csv('heart.csv')
heart.head()
male age education currentSmoker cigsPerDay BPMeds prevalentStroke prevalentHyp diabetes totChol sysBP diaBP BMI heartRate glucose TenYearCHD
0 1 39 4.0 0 0.0 0.0 0 0 0 195.0 106.0 70.0 26.97 80.0 77.0 0
1 0 46 2.0 0 0.0 0.0 0 0 0 250.0 121.0 81.0 28.73 95.0 76.0 0
2 1 48 1.0 1 20.0 0.0 0 0 0 245.0 127.5 80.0 25.34 75.0 70.0 0
3 0 61 3.0 1 30.0 0.0 0 1 0 225.0 150.0 95.0 28.58 65.0 103.0 1
4 0 46 3.0 1 23.0 0.0 0 0 0 285.0 130.0 84.0 23.10 85.0 85.0 0
[5] heart.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4240 entries, 0 to 4239
Data columns (total 16 columns):
Column Non-Null Count Dtype
0 male 4240 non-null int64
1 age 4240 non-null int64
2 education 4135 non-null float64
3 currentSmoker 4240 non-null int64
4 cigsPerDay 4211 non-null float64
5 BPMeds 4187 non-null float64
6 prevalentStroke 4240 non-null int64
7 prevalentHyp 4240 non-null int64
8 diabetes 4240 non-null int64
9 totChol 4190 non-null float64
10 sysBP 4240 non-null float64


Untitled23.ipynb - Colaboratory
heart.ipynb - Colaboratory
colab.research.google.com/drive/1u5qTEPPdwR62YCFMW1q7ZOzLxeGfkIOU#scrollTo=MSo3wqlkVRF8
heart.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
RAM Disk
Editing
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4240 entries, 0 to 4239
Data columns (total 16 columns):
Column Non-Null Count Dtype
0 male 4240 non-null int64
1 age 4240 non-null int64
2 education 4135 non-null float64
3 currentSmoker 4240 non-null int64
4 cigsPerDay 4211 non-null float64
5 BPMeds 4187 non-null float64
6 prevalentStroke 4240 non-null int64
7 prevalentHyp 4240 non-null int64
8 diabetes 4240 non-null int64
9 totChol 4190 non-null float64
10 sysBP 4240 non-null float64
11 diaBP 4240 non-null float64
12 BMI 4221 non-null float64
13 heartRate 4239 non-null float64
14 glucose 3852 non-null float64
15 TenYearCHD 4240 non-null int64
dtypes: float64(9), int64(7)
memory usage: 530.1 KB
[6] heart.shape
(4240, 16)
[7] heart.describe()
male age education currentSmoker cigsPerDay BPMeds prevalentStroke prevalentHyp diabetes totChol sysBP dia

Untitled23.ipynb - Colaboratory
heart.ipynb - Colaboratory
colab.research.google.com/drive/1u5qTEPPdwR62YCFMW1q7ZOzLxeGfkIOU#scrollTo=MSo3wqlkVRF8
heart.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
RAM Disk
Editing
[7] heart.describe()
male age education currentSmoker cigsPerDay BPMeds prevalentStroke prevalentHyp diabetes totChol sysBP dia
count 4240.000000 4240.000000 4135.000000 4240.000000 4211.000000 4187.000000 4240.000000 4240.000000 4240.000000 4190.000000 4240.000000 4240.000
mean 0.429245 49.580189 1.979444 0.494104 9.005937 0.029615 0.005896 0.310613 0.025708 236.699523 132.354599 82.8977
std 0.495027 8.572942 1.019791 0.500024 11.922462 0.169544 0.076569 0.462799 0.158280 44.591284 22.033300 11.9103
min 0.000000 32.000000 1.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 107.000000 83.500000 48.000
25% 0.000000 42.000000 1.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 206.000000 117.000000 75.000
50% 0.000000 49.000000 2.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 234.000000 128.000000 82.000
75% 1.000000 56.000000 3.000000 1.000000 20.000000 0.000000 0.000000 1.000000 0.000000 263.000000 144.000000 90.000
max 1.000000 70.000000 4.000000 1.000000 70.000000 1.000000 1.000000 1.000000 1.000000 696.000000 295.000000 142.500
[8] heart.isnull().sum()
male 0
age 0
education 105
currentSmoker 0
cigsPerDay 29
BPMeds 53
prevalentStroke 0
prevalentHyp 0
diabetes 0
totChol 50


Untitled23.ipynb - Colaboratory
heart.ipynb - Colaboratory
colab.research.google.com/drive/1u5qTEPPdwR62YCFMW1q7ZOzLxeGfkIOU#scrollTo=MSo3wqlkVRF8
heart.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
RAM Disk
Editing
[7] heart.describe()
Smoker cigsPerDay BPMeds prevalentStroke prevalentHyp diabetes totChol sysBP diaBP BMI heartRate glucose TenYearCHD
000000 4211.000000 4187.000000 4240.000000 4240.000000 4240.000000 4190.000000 4240.000000 4240.000000 4221.000000 4239.000000 3852.000000 4240.000000
494104 9.005937 0.029615 0.005896 0.310613 0.025708 236.699523 132.354599 82.897759 25.800801 75.878981 81.963655 0.151887
500024 11.922462 0.169544 0.076569 0.462799 0.158280 44.591284 22.033300 11.910394 4.079840 12.025348 23.954335 0.358953
000000 0.000000 0.000000 0.000000 0.000000 0.000000 107.000000 83.500000 48.000000 15.540000 44.000000 40.000000 0.000000
000000 0.000000 0.000000 0.000000 0.000000 0.000000 206.000000 117.000000 75.000000 23.070000 68.000000 71.000000 0.000000
000000 0.000000 0.000000 0.000000 0.000000 0.000000 234.000000 128.000000 82.000000 25.400000 75.000000 78.000000 0.000000
000000 20.000000 0.000000 0.000000 1.000000 0.000000 263.000000 144.000000 90.000000 28.040000 83.000000 87.000000 0.000000
000000 70.000000 1.000000 1.000000 1.000000 1.000000 696.000000 295.000000 142.500000 56.800000 143.000000 394.000000 1.000000
heart.isnull().sum()
male 0
age 0
education 105
currentSmoker 0
cigsPerDay 29
BPMeds 53
prevalentStroke 0
prevalentHyp 0
diabetes 0
totChol 50

Untitled23.ipynb - Colaboratory
heart.ipynb - Colaboratory
colab.research.google.com/drive/1u5qTEPPdwR62YCFMW1q7ZOzLxeGfkIOU#scrollTo=MSo3wqlkVRF8
heart.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
RAM Disk
Editing
[8] heart.isnull().sum()
male 0
age 0
education 105
currentSmoker 0
cigsPerDay 29
BPMeds 53
prevalentStroke 0
prevalentHyp 0
diabetes 0
totChol 50
sysBP 0
diaBP 0
BMI 19
heartRate 1
glucose 388
TenYearCHD 0
dtype: int64
[9] heart_fill = heart
[10] heart_fill["cigsPerDay"].fillna(heart_fill.groupby("currentSmoker")["cigsPerDay"].transform("mean"), inplace=True)
[11] heart_fill["BPMeds"].fillna(heart_fill.groupby("prevalentHyp")["BPMeds"].transform('median'), inplace=True)
[12] heart_fill["totChol"].fillna(heart_fill.groupby("prevalentStroke")["totChol"].transform("mean"), inplace=True)
[13] heart_fill["BMI"].fillna(heart_fill.groupby("age")["BMI"].transform("mean"), inplace=True)



	male	age	education	currentSmoker	cigsPerDay	BPMeds	prevalentStroke	prevalentHyp	diabetes	totChol	sysBP	diaBP	BMI	heartRate	glucose	TenYearCHD
0	1	39	4.0	0	0.0	0.0	0	0	0	195.0	106.0	70.0	26.97	80.0	77.0	0
1	0	46	2.0	0	0.0	0.0	0	0	0	250.0	121.0	81.0	28.73	95.0	76.0	0
2	1	48	1.0	1	20.0	0.0	0	0	0	245.0	127.5	80.0	25.34	75.0	70.0	0
3	0	61	3.0	1	30.0	0.0	0	1	0	225.0	150.0	95.0	28.58	65.0	103.0	1
4	0	46	3.0	1	23.0	0.0	0	0	0	285.0	130.0	84.0	23.10	85.0	85.0	0

Untitled23.ipynb - Colaboratory | heart.ipynb - Colaboratory

colab.research.google.com/drive/1u5qTEPPdwR62YCFMW1q7ZOzLxeGfkIOU#scrollTo=MSo3wqIkVRF8

heart.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk Editing

```
[18] heart_fill.head()
```

	male	age	education	currentSmoker	cigsPerDay	BPMeds	prevalentStroke	prevalentHyp	diabetes	totChol	sysBP	diaBP	BMI	heartRate	glucose	TenYearCHD
0	1	39	4.0	0	0.0	0.0	0	0	0	195.0	106.0	70.0	26.97	80.0	77.0	0
1	0	46	2.0	0	0.0	0.0	0	0	0	250.0	121.0	81.0	28.73	95.0	76.0	0
2	1	48	1.0	1	20.0	0.0	0	0	0	245.0	127.5	80.0	25.34	75.0	70.0	0
3	0	61	3.0	1	30.0	0.0	0	1	0	225.0	150.0	95.0	28.58	65.0	103.0	1
4	0	46	3.0	1	23.0	0.0	0	0	0	285.0	130.0	84.0	23.10	85.0	85.0	0

Check For Correlation between attributes.

```
sns.heatmap(heart_fill.corr())
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f98b2e6b5c0>
```

Untitled23.ipynb - Colaboratory | heart.ipynb - Colaboratory

colab.research.google.com/drive/1u5qTEPPdwR62YCFMW1q7ZOzLxeGfkIOU#scrollTo=MSo3wqIkVRF8

heart.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

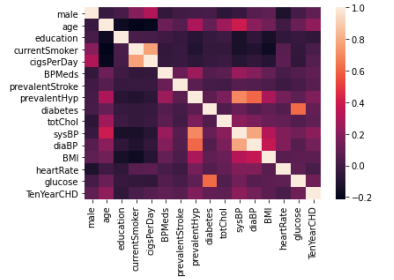
+ Code + Text

RAM Disk Editing

Check For Correlation between attributes.

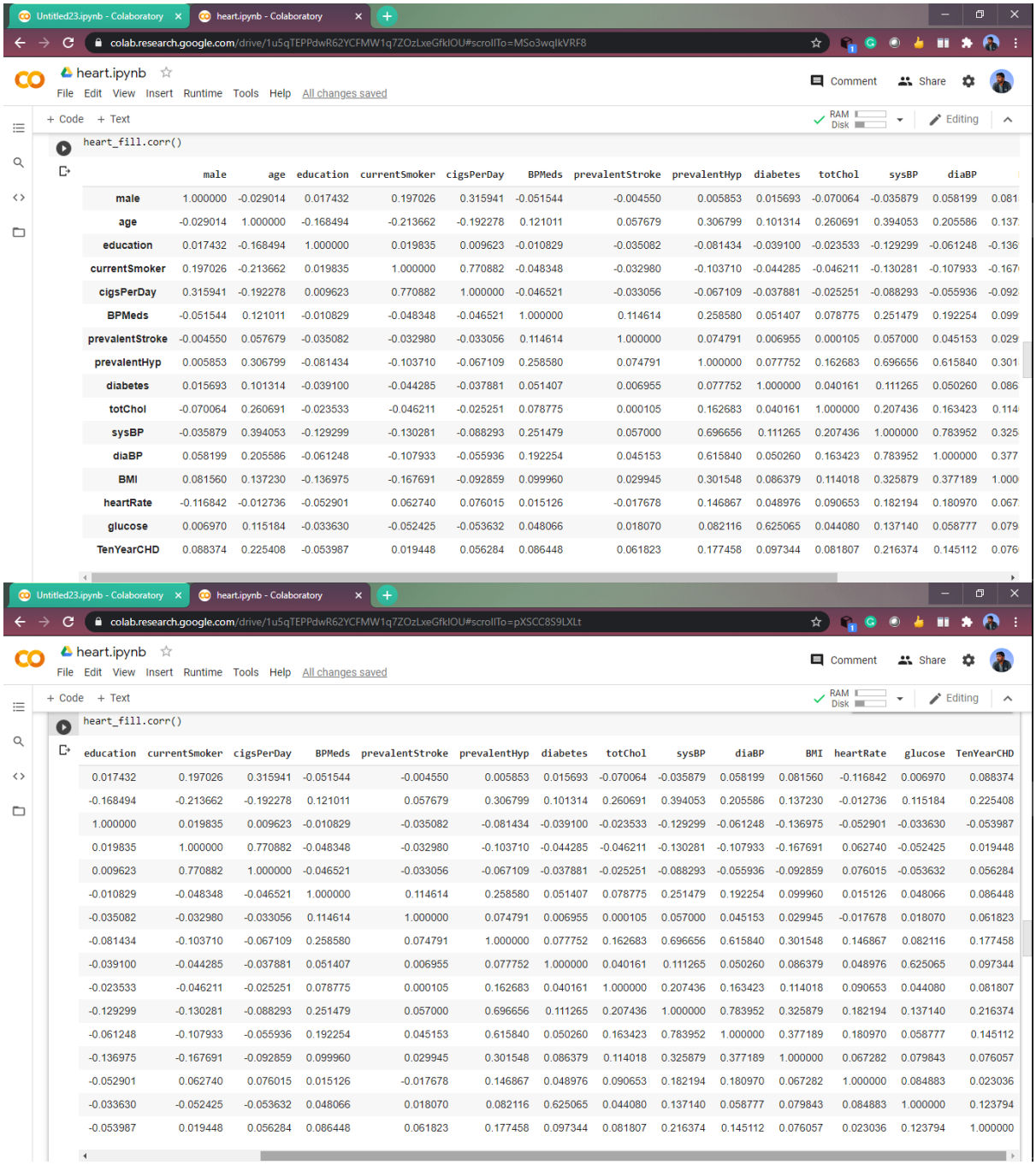
```
[19] sns.heatmap(heart_fill.corr())
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f98b2e6b5c0>
```




```
[20] heart_fill.corr()
```

	male	age	education	currentSmoker	cigsPerDay	BPMeds	prevalentStroke	prevalentHyp	diabetes	totChol	sysBP	diaBP	
male	1.000000	-0.029014	0.017432	0.197026	0.315941	-0.051544	-0.004550	0.005853	0.015693	-0.070064	-0.035879	0.058199	0.081



	male	age	education	currentSmoker	cigsPerDay	BPMeds	prevalentStroke	prevalentHyp	diabetes	totChol	sysBP	diaBP	
male	1.000000	-0.029014	0.017432	0.197026	0.315941	-0.051544	-0.004550	0.005853	0.015693	-0.070064	-0.035879	0.058199	0.081
age	-0.029014	1.000000	-0.168494	-0.213662	-0.192278	0.121011	0.057679	0.306799	0.101314	0.260691	0.394053	0.205586	0.137
education	0.017432	-0.168494	1.000000	0.019835	0.009623	-0.010829	-0.035082	-0.081434	-0.039100	-0.023533	-0.129299	-0.061248	-0.136
currentSmoker	0.197026	-0.213662	0.019835	1.000000	0.770882	-0.048348	-0.032980	-0.103710	-0.044285	-0.046211	-0.130281	-0.107933	-0.167
cigsPerDay	0.315941	-0.192278	0.009623	0.770882	1.000000	-0.046521	-0.033056	-0.067109	-0.037881	-0.025251	-0.088293	-0.055936	-0.092
BPMeds	-0.051544	0.121011	-0.010829	-0.048348	-0.046521	1.000000	0.114614	0.258580	0.051407	0.078775	0.251479	0.192254	0.099
prevalentStroke	-0.004550	0.057679	-0.035082	-0.032980	-0.033056	0.114614	1.000000	0.074791	0.006955	0.000105	0.057000	0.045153	0.029
prevalentHyp	0.005853	0.306799	-0.081434	-0.103710	-0.067109	0.258580	0.074791	1.000000	0.077752	0.162683	0.696656	0.615840	0.301
diabetes	0.015693	0.101314	-0.039100	-0.044285	-0.037881	0.051407	0.006955	0.077752	1.000000	0.040161	0.111265	0.050260	0.086
totChol	-0.070064	0.260691	-0.023533	-0.046211	-0.025251	0.078775	0.000105	0.162683	0.040161	1.000000	0.207436	0.163423	0.114
sysBP	-0.035879	0.394053	-0.129299	-0.130281	-0.088293	0.251479	0.057000	0.696656	0.111265	0.207436	1.000000	0.783952	0.325
diaBP	0.058199	0.205586	-0.061248	-0.107933	-0.055936	0.192254	0.045153	0.615840	0.050260	0.163423	0.783952	1.000000	0.377
BMI	0.081560	0.137230	-0.136975	-0.167691	-0.092859	0.099960	0.029945	0.301548	0.086379	0.114018	0.325879	0.377189	1.000
heartRate	-0.116842	-0.012736	-0.052901	0.062740	0.076015	0.015126	-0.017678	0.146867	0.048976	0.090653	0.182194	0.180970	0.067
glucose	0.006970	0.115184	-0.033630	-0.052425	-0.053632	0.048066	0.018070	0.082116	0.625065	0.044080	0.137140	0.058777	0.079
TenYearCHD	0.088374	0.225408	-0.053987	0.019448	0.056284	0.086448	0.061823	0.177458	0.097344	0.081807	0.216374	0.145112	0.076

education	currentSmoker	cigsPerDay	BPMeds	prevalentStroke	prevalentHyp	diabetes	totChol	sysBP	diaBP	BMI	heartRate	glucose	TenYearCHD
0.017432	0.197026	0.315941	-0.051544	-0.004550	0.005853	0.015693	-0.070064	-0.035879	0.058199	0.081560	-0.116842	0.006970	0.088374
-0.168494	-0.213662	-0.192278	0.121011	0.057679	0.306799	0.101314	0.260691	0.394053	0.205586	0.137230	-0.012736	0.115184	0.225408
1.000000	0.019835	0.009623	-0.010829	-0.035082	-0.081434	-0.039100	-0.023533	-0.129299	-0.061248	-0.136975	-0.052901	-0.033630	-0.053987
0.019835	1.000000	0.770882	-0.048348	-0.032980	-0.103710	-0.044285	-0.046211	-0.130281	-0.107933	-0.167691	0.062740	-0.052425	0.019448
0.009623	0.770882	1.000000	-0.046521	-0.033056	-0.067109	-0.037881	-0.025251	-0.088293	-0.055936	-0.092859	0.076015	-0.053632	0.056284
-0.010829	-0.048348	-0.046521	1.000000	0.114614	0.258580	0.051407	0.078775	0.251479	0.192254	0.099960	0.015126	0.048066	0.086448
-0.035082	-0.032980	-0.033056	0.114614	1.000000	0.074791	0.006955	0.000105	0.057000	0.045153	0.029945	-0.017678	0.018070	0.061823
-0.081434	-0.103710	-0.067109	0.258580	0.074791	1.000000	0.077752	0.162683	0.696656	0.615840	0.301548	0.146867	0.082116	0.177458
-0.039100	-0.044285	-0.037881	0.051407	0.006955	0.077752	1.000000	0.040161	0.111265	0.050260	0.086379	0.048976	0.625065	0.097344
-0.023533	-0.046211	-0.025251	0.078775	0.000105	0.162683	0.040161	1.000000	0.207436	0.163423	0.114018	0.090653	0.044080	0.081807
-0.129299	-0.130281	-0.088293	0.251479	0.057000	0.696656	0.111265	0.207436	1.000000	0.783952	0.325879	0.182194	0.137140	0.216374
-0.061248	-0.107933	-0.055936	0.192254	0.045153	0.615840	0.050260	0.163423	0.783952	1.000000	0.377189	0.180970	0.058777	0.145112
-0.136975	-0.167691	-0.092859	0.099960	0.029945	0.301548	0.086379	0.114018	0.325879	0.377189	1.000000	0.067282	0.079843	0.076057
-0.052901	0.062740	0.076015	0.015126	-0.017678	0.146867	0.048976	0.090653	0.182194	0.180970	0.067282	1.000000	0.084883	0.023036
-0.033630	-0.052425	-0.053632	0.048066	0.018070	0.082116	0.625065	0.044080	0.137140	0.058777	0.079843	0.084883	1.000000	0.123794
-0.053987	0.019448	0.056284	0.086448	0.061823	0.177458	0.097344	0.081807	0.216374	0.145112	0.076057	0.023036	0.123794	1.000000

Untitled23.ipynb - Colaboratory | heart.ipynb - Colaboratory

colab.research.google.com/drive/1u5qTEPPdwR62YCFMW1q7ZOzLxeGfkIOU#scrollTo=pXSCC8S9LXLt

heart.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

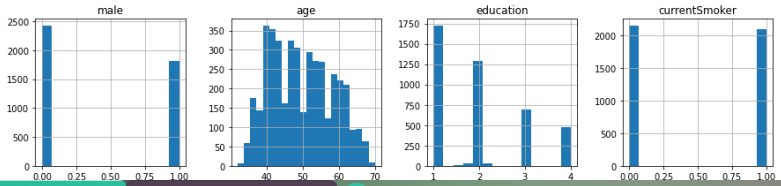
+ Code + Text

RAM Disk Editing

Visualize the data

```
heart_fill.hist(bins='auto', figsize=(15,15))
```

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7f98afff048>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f98affaf2e8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f98aff63550>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f98aff977b8>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7f98aff4aa20>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f98afefdc88>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f98afeb4ef0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f98afe73160>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7f98afe731d0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f98afe5c668>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f98afe0f8d0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f98afdc3b38>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7f98afd77da0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f98afd2bfd0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f98afceb2b0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f98afd1e518>]],
      dtype=object)
```



Untitled23.ipynb - Colaboratory | heart.ipynb - Colaboratory

colab.research.google.com/drive/1u5qTEPPdwR62YCFMW1q7ZOzLxeGfkIOU#scrollTo=pXSCC8S9LXLt

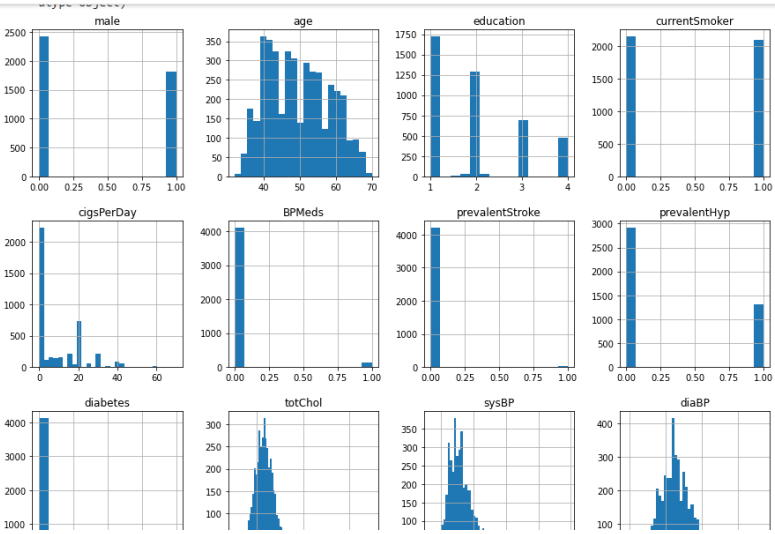
heart.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk Editing



Untitled23.ipynb - Colaboratory | heart.ipynb - Colaboratory

colab.research.google.com/drive/1u5qTEPPdwR62YCFMW1q7ZOzLxeGfkIOU#scrollTo=pXSCC8S9LXLt

heart.ipynb

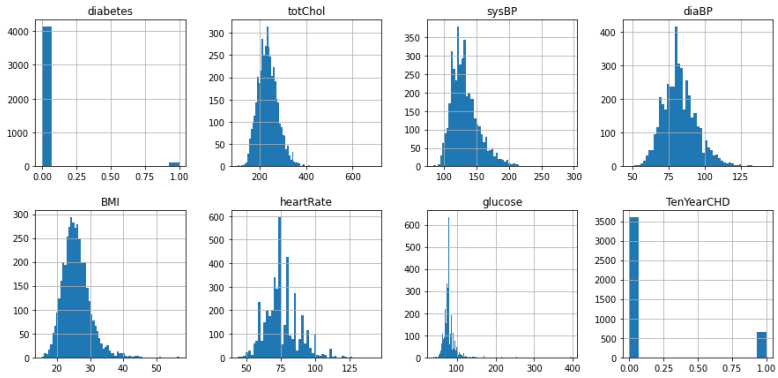
File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk Editing

[21]




```
[24] X=heart_fill.drop(['TenYearCHD'], axis=1)
     y=heart_fill['TenYearCHD']
```

```
# Seperating training and testing data
from sklearn.model_selection import train_test_split
```

+ Code + Text

Untitled23.ipynb - Colaboratory | heart.ipynb - Colaboratory

colab.research.google.com/drive/1u5qTEPPdwR62YCFMW1q7ZOzLxeGfkIOU#scrollTo=pXSCC8S9LXLt

heart.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk Editing

```
[24] X=heart_fill.drop(['TenYearCHD'], axis=1)
     y=heart_fill['TenYearCHD']
```

```
[25] # Seperating training and testing data
     from sklearn.model_selection import train_test_split
```

```
[26] X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33)
```

```
[27] from sklearn.linear_model import LogisticRegression
```

```
[32] model=LogisticRegression(max_iter=5000)
```

```
[33] model.fit(X_train,y_train)

     LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                        intercept_scaling=1, l1_ratio=None, max_iter=5000,
                        multi_class='auto', n_jobs=None, penalty='l2',
                        random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                        warm_start=False)
```

```
[34] pred=model.predict(X_test)
```

```
[36] from sklearn import metrics
```

```
[37] print(metrics.classification_report(y_pred=pred, y_true=y_test))
```

Untitled23.ipynb - Colaboratory | heart.ipynb - Colaboratory

colab.research.google.com/drive/1u5qTEPPdwR62YCFMW1q7ZOzLxeGfkIOU#scrollTo=pXSCC8S9LXLt

heart.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share Editing

+ Code + Text

RAM Disk

```
[34] pred=model.predict(X_test)
```

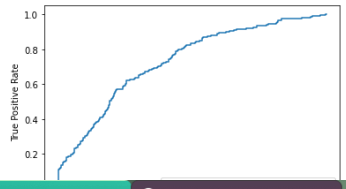
```
[36] from sklearn import metrics
```

```
[37] print(metrics.classification_report(y_pred=pred, y_true=y_test))
```

```
              precision    recall  f1-score   support

           0       0.85      1.00      0.92      1174
           1       0.84      0.09      0.17       226

    accuracy                           0.85      1400
   macro avg       0.85      0.54      0.54      1400
weighted avg       0.85      0.85      0.80      1400
```

```
[39] metrics.plot_roc_curve(model,X_test,y_test)
```

```
<sklearn.metrics._plot.roc_curve.RocCurveDisplay at 0x7f98aa464160>
```



Untitled23.ipynb - Colaboratory | heart.ipynb - Colaboratory

colab.research.google.com/drive/1u5qTEPPdwR62YCFMW1q7ZOzLxeGfkIOU#scrollTo=pXSCC8S9LXLt

heart.ipynb

File Edit View Insert Runtime Tools Help All changes saved

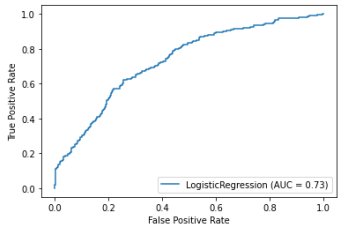
Comment Share Editing

+ Code + Text

RAM Disk

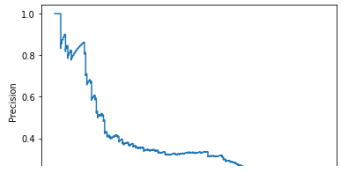
[39] metrics.plot_roc_curve(model,X_test,y_test)

<sklearn.metrics._plot.roc_curve.RocCurveDisplay at 0x7f98aa464160>




```
metrics.plot_precision_recall_curve(model,X_test,y_test)
```

```
<sklearn.metrics._plot.precision_recall_curve.PrecisionRecallDisplay at 0x7f98aa3e65f8>
```



Untitled23.ipynb - Colaboratory | heart.ipynb - Colaboratory

colab.research.google.com/drive/1u5qTEPPdwR62YCFMW1q7ZOzLxeGfklOU#scrollTo=pXSCC8S9LXLt

heart.ipynb

File Edit View Insert Runtime Tools Help All changes saved

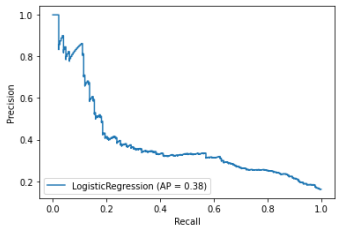
Comment Share

+ Code + Text

RAM Disk Editing

```
[40] metrics.plot_precision_recall_curve(model,X_test,y_test)
```

```
<sklearn.metrics._plot.precision_recall_curve.PrecisionRecallDisplay at 0x7f98aa3e65f8>
```




```
[41] from sklearn.metrics import accuracy_score
```

```
[43] print(accuracy_score(y_test,pred))
```

```
0.8507142857142858
```

```
[ ]
```

Abhinay Bhatt - 179301009 - CSE A

To perform K-Means Clustering for Car Dataset

```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
from sklearn.cluster import KMeans
```

```
data = pd.read_csv('cars.csv')
data.head()
```

	mpg	cylinders	cubicinches	hp	weightlbs	time-to-60	year	brand
0	14.0	8	350	165	4209	12	1972	US.
1	31.9	4	89	71	1925	14	1980	Europe.
2	17.0	8	302	140	3449	11	1971	US.
3	15.0	8	400	150	3761	10	1971	US.
4	30.5	4	98	63	2051	17	1978	US.

```
data.isnull().sum()
```

```
mpg            0
 cylinders     0
 cubicinches   0
 hp            0
 weightlbs     0
 time-to-60    0
 year          0
 brand         0
dtype: int64
```

```
brand=data[' brand'].str.strip()
data=data.drop(labels=' brand',axis=1)
```

```
data=data.replace(r'^\s*$', np.nan, regex=True)
data=data.astype(dtype=float, errors='ignore')
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 261 entries, 0 to 260
Data columns (total 7 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   mpg           261 non-null    float64
 1    cylinders    261 non-null    float64
 2    cubicinches  259 non-null    float64
 3    hp           261 non-null    float64
 4    weightlbs    258 non-null    float64
 5    time-to-60   261 non-null    float64
 6    year         261 non-null    float64
dtypes: float64(7)
memory usage: 14.4 KB
```

```
from sklearn.impute import SimpleImputer
imputer = SimpleImputer(missing_values=np.nan, strategy='mean')
data=pd.DataFrame(data=imputer.fit_transform(data), columns=data.columns)
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 261 entries, 0 to 260
Data columns (total 7 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   mpg           261 non-null    float64
 1    cylinders    261 non-null    float64
 2    cubicinches  261 non-null    float64
 3    hp           261 non-null    float64
 4    weightlbs    261 non-null    float64
 5    time-to-60   261 non-null    float64
 6    year         261 non-null    float64
dtypes: float64(7)
memory usage: 14.4 KB
```

```
data['brand']=brand
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 261 entries, 0 to 260
Data columns (total 8 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   mpg           261 non-null    float64
 1    cylinders    261 non-null    float64
 2    cubicinches  261 non-null    float64
 3    hp           261 non-null    float64
 4    weightlbs    261 non-null    float64
 5    time-to-60   261 non-null    float64
```

```
 6    year         261 non-null    float64
 7   brand         261 non-null    object
dtypes: float64(7), object(1)
memory usage: 16.4+ KB
```

```
print(data.iloc[:,7])
```

```
0         US.
1      Europe.
2         US.
3         US.
4         US.
        ...
256       US.
257     Japan.
258       US.
259       US.
260       US.
Name: brand, Length: 261, dtype: object
```

```
#
X=data.iloc[:,0:8].values
X
```

```
array([[14.0, 8.0, 350.0, ..., 12.0, 1972.0, 'US.'],
       [31.9, 4.0, 89.0, ..., 14.0, 1980.0, 'Europe.'],
       [17.0, 8.0, 302.0, ..., 11.0, 1971.0, 'US.'],
       ...,
       [22.0, 6.0, 232.0, ..., 15.0, 1983.0, 'US.'],
       [18.0, 6.0, 232.0, ..., 16.0, 1972.0, 'US.'],
       [22.0, 6.0, 250.0, ..., 15.0, 1977.0, 'US.']], dtype=object)
```

```
# String label int/float
from sklearn.preprocessing import LabelEncoder

# this is for country labeling
cont=LabelEncoder()

# Now apply column first in this labelEn
X[:,7]=cont.fit_transform(X[:,7])
```

```
X
```

```
array([[14.0, 8.0, 350.0, ..., 12.0, 1972.0, 2],
       [31.9, 4.0, 89.0, ..., 14.0, 1980.0, 0],
       [17.0, 8.0, 302.0, ..., 11.0, 1971.0, 2],
       ...,
       [22.0, 6.0, 232.0, ..., 15.0, 1983.0, 2],
       [18.0, 6.0, 232.0, ..., 16.0, 1972.0, 2],
       [22.0, 6.0, 250.0, ..., 15.0, 1977.0, 2]], dtype=object)
```

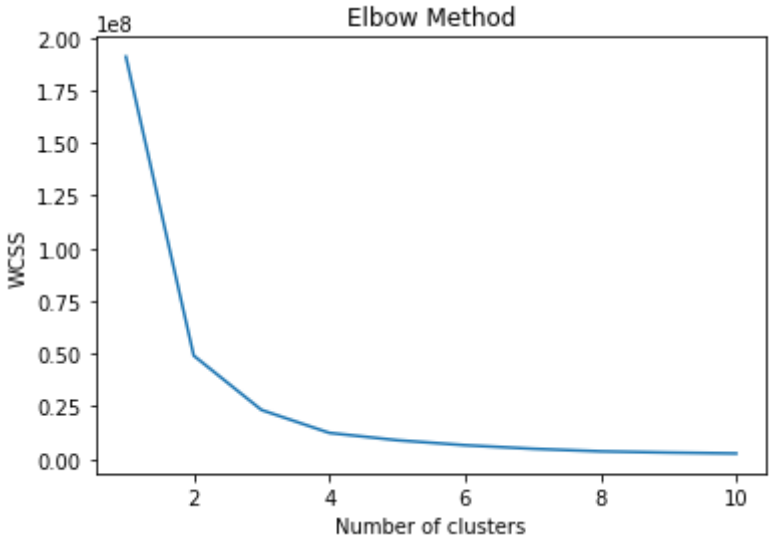
```
# Now encoding First Column i.e. making subcolumn of first column
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer

columnTransformer = ColumnTransformer([('encoder',
                                        OneHotEncoder(),
                                        [-1])],
                                      remainder='passthrough')

X = np.array(columnTransformer.fit_transform(X), dtype = float)
```

```
X[2,:]
```

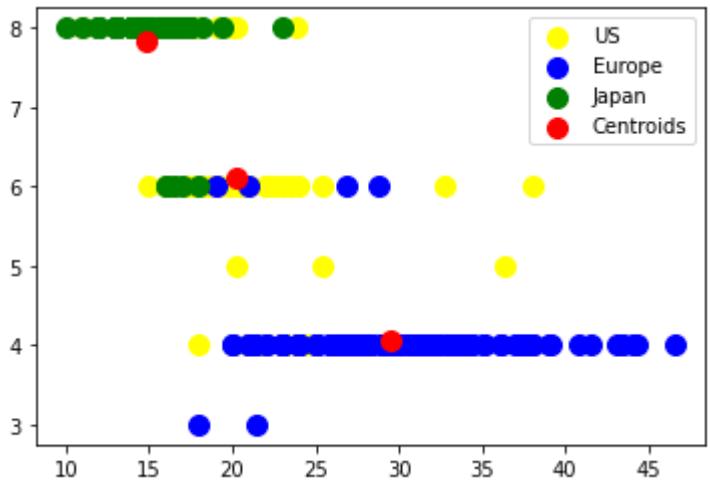
```
array([0.000e+00, 0.000e+00, 1.000e+00, 1.700e+01, 8.000e+00, 3.020e+02,
       1.400e+02, 3.449e+03, 1.100e+01, 1.971e+03])
```

```
X=X.astype(np.float)
```

```
wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, init='k-means++', max_iter=300, n_init=10, random_state=
    kmeans.fit(data)
    wcss.append(kmeans.inertia_)
plt.plot(range(1, 11), wcss)
plt.title('Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
```



```
kmeans = KMeans(n_clusters=3, init='k-means++', max_iter=100, random_state=10)
pred = kmeans.fit_predict(data)

data=np.array(data)
plt.scatter(data[pred == 0,0], data[pred == 0,1], s=100, c='yellow', label='US')
plt.scatter(data[pred == 1,0], data[pred == 1,1], s=100, c='blue', label='Europe')
plt.scatter(data[pred == 2,0], data[pred == 2,1], s=100, c='green', label='Japan')

plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s=100, c='red',
plt.legend()
plt.show()
```



Abhinay Bhatt - 179301009 - CSE A

To perform k-Nearest Neighbours Algorithm any dataset.

```
# loading iris data
from sklearn.datasets import load_iris
```

```
iris_data=load_iris()
```

```
dir(iris_data
   )
```

```
['DESCR', 'data', 'feature_names', 'filename', 'target', 'target_names']
```

```
features=iris_data.data
label=iris_data.target
```

```
# seperate data into training and testing
from sklearn.model_selection import train_test_split
```

Train-Data = 70%

```
train_data,test_data,train_label,test_label=train_test_split(features,label,test_size=.3)
```

```
# import KNN classifier
from sklearn.neighbors import KNeighborsClassifier
```

```
# KNN classifier with k=3
kclf=KNeighborsClassifier(n_neighbors=3)
```

```
# KNN classifier with k=5
```

```
kclf2=KNeighborsClassifier(n_neighbors=5)


# KNN classifier with k=7
kclf3=KNeighborsClassifier(n_neighbors=7)


# train data using KNN
ktrained=kclf.fit(train_data,train_label)
ktrained1b=kclf2.fit(train_data,train_label)
ktrained1c=kclf3.fit(train_data,train_label)


#Predict using KNN
kprediction=ktrained.predict(test_data)
kprediction1b=ktrained1b.predict(test_data)
kprediction1c=ktrained1c.predict(test_data)


#to check accuracy
from sklearn.metrics import accuracy_score


# Check Accuracy of KNN Prediction
kacc=accuracy_score(kprediction,test_label)
kacc1b=accuracy_score(kprediction1b,test_label)
kacc1c=accuracy_score(kprediction1c,test_label)


kacc
```

```
0.9777777777777777
```

```
kacc1b
```

```
0.9555555555555556
```

```
kacc1c
```

```
0.9555555555555556
```

Train-Data = 80%

```
# Separate data with 80% train data
train_data2,test_data2,train_label2,test_label2=train_test_split(features,label,test_size=


# Train using KNN
ktrained2=kclf.fit(train_data2,train_label2)
ktrained2b=kclf2.fit(train_data2,train_label2)
ktrained2c=kclf3.fit(train_data2,train_label2)
```

```
# Predict using Prediction
kprediction2=ktrained2.predict(test_data2)
kprediction2b=ktrained2b.predict(test_data2)
kprediction2c=ktrained2c.predict(test_data2)


# Check Accuracy of KNN prediction
kacc2=accuracy_score(kprediction2,test_label2)
kacc2b=accuracy_score(kprediction2b,test_label2)
kacc2c=accuracy_score(kprediction2c,test_label2)


kacc2
```

```
0.9333333333333333
```

```
kacc2b
```

```
0.9
```

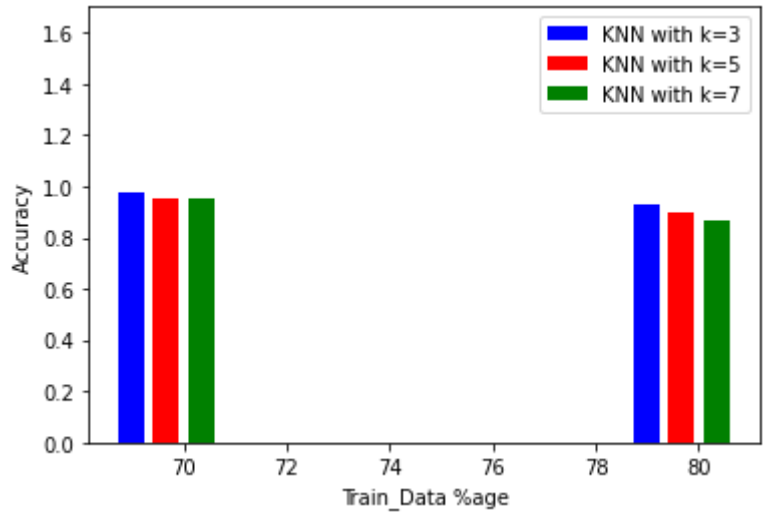
```
kacc2c
```

```
0.8666666666666667
```

```
# plotting graph
import matplotlib.pyplot as plt


plt.ylim(0,1.7)

plt.xlabel("Train_Data %age")
plt.ylabel("Accuracy")
x1=70
x2=80

yk1=kacc
yk1b=kacc1b
yk1c=kacc1c

yk2=kacc2
yk2b=kacc2b
yk2c=kacc2c

width=.5
plt.bar(x1-2*width,yk1,width,color="blue",label="KNN with k=3")
plt.bar(x1-width/2-.1,yk1b,width,color="red",label="KNN with k=5")
plt.bar(x1+width/2+.1,yk1c,width,color="green",label="KNN with k=7")

plt.bar(x2-2*width,yk2,width,color="blue")
plt.bar(x2-width/2-.1,yk2b,width,color="red")
plt.bar(x2+width/2+.1,yk2c,width,color="green")
```

```
plt.legend()
```

```
<matplotlib.legend.Legend at 0x7fda7d8d9278>
```



Abhinay Bhatt - 179301009 - CSE A

To perform Time Series Analysis on Superstore sales data.

```
import warnings
import itertools
import numpy as np
import matplotlib.pyplot as plt
warnings.filterwarnings("ignore")
plt.style.use('fivethirtyeight')
import pandas as pd
import statsmodels.api as sm
import matplotlib
matplotlib.rcParams['axes.labelsize'] = 14
matplotlib.rcParams['xtick.labelsize'] = 12
matplotlib.rcParams['ytick.labelsize'] = 12
matplotlib.rcParams['text.color'] = 'k'
```



```
df = pd.read_excel("/content/Sample - Superstore.xls")
furniture = df.loc[df['Category'] == 'Furniture']
```

```
furniture['Order Date'].min(), furniture['Order Date'].max()
```

```
(Timestamp('2014-01-06 00:00:00'), Timestamp('2017-12-30 00:00:00'))
```

Data Preprocessing

```
cols = ['Row ID', 'Order ID', 'Ship Date', 'Ship Mode', 'Customer ID', 'Customer Name', 'S
furniture.drop(cols, axis=1, inplace=True)
furniture = furniture.sort_values('Order Date')
furniture.isnull().sum()
```

```
Order Date    0
Sales         0
dtype: int64
```

```
furniture = furniture.groupby('Order Date')['Sales'].sum().reset_index()
```

```
furniture = furniture.set_index('Order Date')
furniture.index
```

```
DatetimeIndex(['2014-01-06', '2014-01-07', '2014-01-10', '2014-01-11',
               '2014-01-13', '2014-01-14', '2014-01-16', '2014-01-19',
               '2014-01-20', '2014-01-21',
               ...
```

```
               '2017-12-18', '2017-12-19', '2017-12-21', '2017-12-22',
               '2017-12-23', '2017-12-24', '2017-12-25', '2017-12-28',
               '2017-12-29', '2017-12-30'],
              dtype='datetime64[ns]', name='Order Date', length=889, freq=None)
```

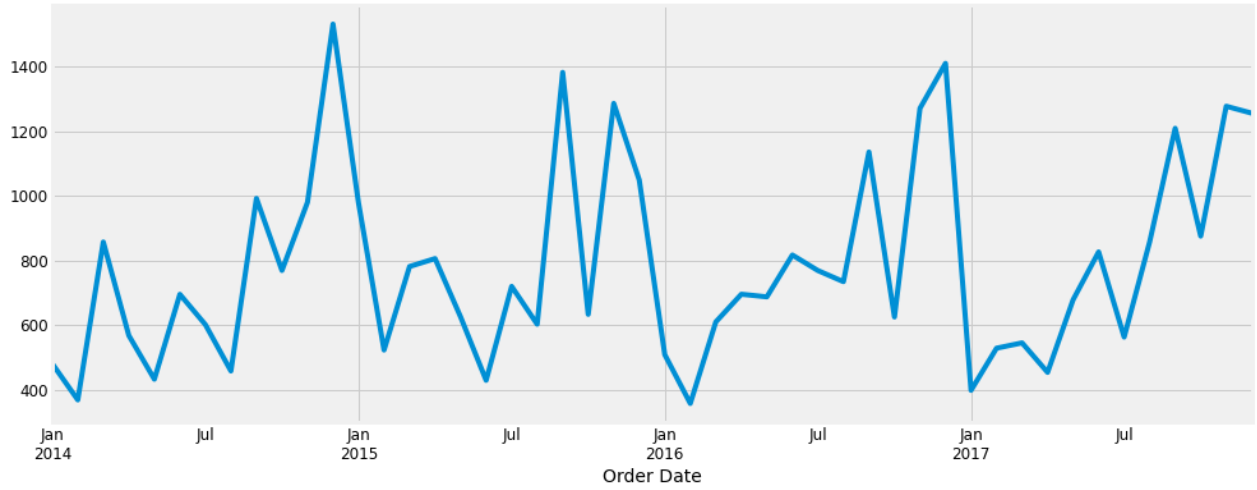
```
y = furniture['Sales'].resample('MS').mean()
```

```
y['2017':]
```

```
Order Date
2017-01-01     397.602133
2017-02-01     528.179800
2017-03-01     544.672240
2017-04-01     453.297905
2017-05-01     678.302328
2017-06-01     826.460291
2017-07-01     562.524857
2017-08-01     857.881889
2017-09-01    1209.508583
2017-10-01     875.362728
2017-11-01    1277.817759
2017-12-01    1256.298672
Freq: MS, Name: Sales, dtype: float64
```

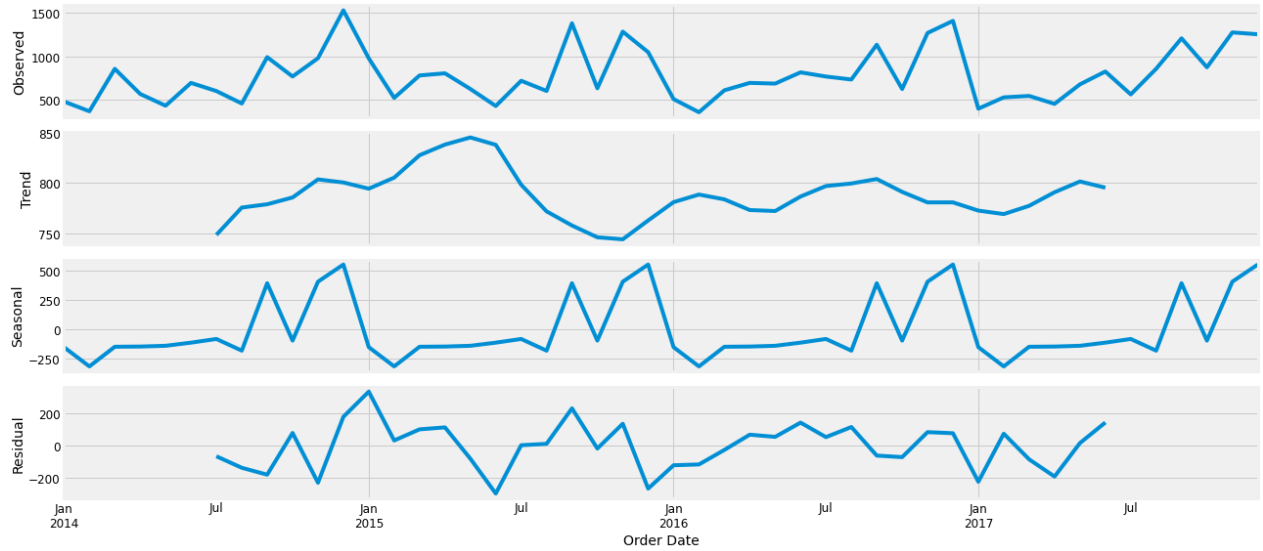
Visualizing the data

```
y.plot(figsize=(15, 6))
plt.show()
```




```
from pylab import rcParams
```

```
rcParams['figure.figsize'] = 18, 8
decomposition = sm.tsa.seasonal_decompose(y, model='additive')
fig = decomposition.plot()
plt.show()
```



Time series forecasting with ARIMA

```
p = d = q = range(0, 2)
pdq = list(itertools.product(p, d, q))
seasonal_pdq = [(x[0], x[1], x[2], 12) for x in list(itertools.product(p, d, q))]
print('Examples of parameter combinations for Seasonal ARIMA...')
print('SARIMAX: {} x {}'.format(pdq[1], seasonal_pdq[1]))
print('SARIMAX: {} x {}'.format(pdq[1], seasonal_pdq[2]))
print('SARIMAX: {} x {}'.format(pdq[2], seasonal_pdq[3]))
print('SARIMAX: {} x {}'.format(pdq[2], seasonal_pdq[4]))
```

```
Examples of parameter combinations for Seasonal ARIMA...
SARIMAX: (0, 0, 1) x (0, 0, 1, 12)
SARIMAX: (0, 0, 1) x (0, 1, 0, 12)
SARIMAX: (0, 1, 0) x (0, 1, 1, 12)
SARIMAX: (0, 1, 0) x (1, 0, 0, 12)
```

```
for param in pdq:
```

```
for param_seasonal in seasonal_pdq:
    try:
        mod = sm.tsa.statespace.SARIMAX(y,order=param,seasonal_order=param_seasonal,en
        results = mod.fit()
        print('ARIMA{}x{}12 - AIC:{}'.format(param, param_seasonal, results.aic))
    except:
        continue
```

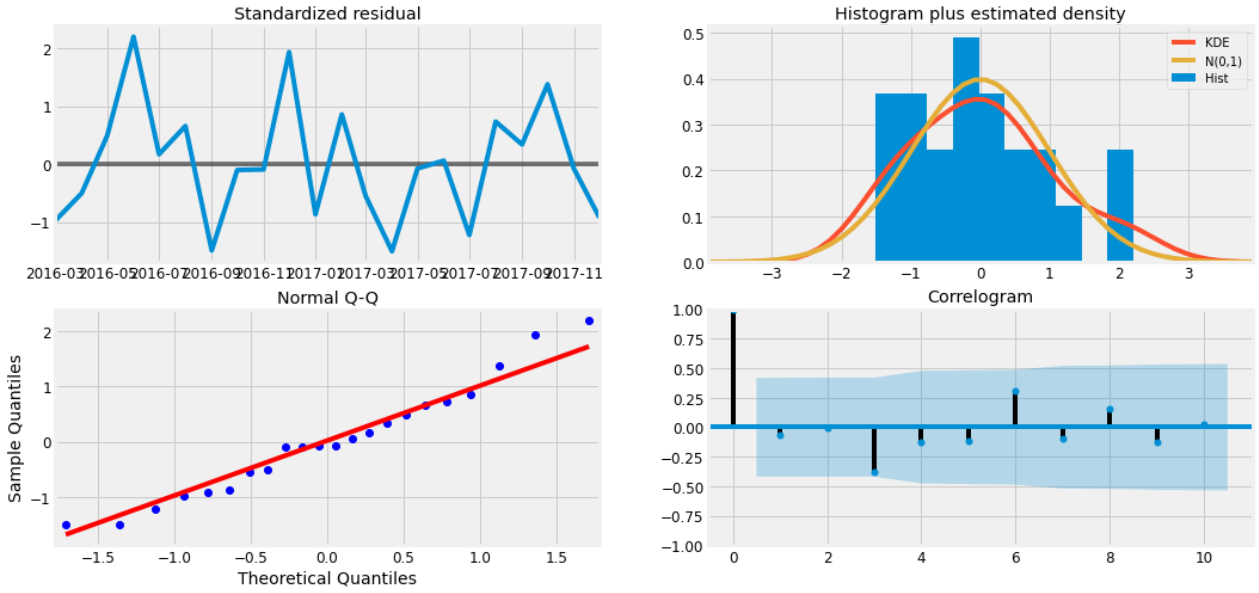
```
ARIMA(0, 0, 0)x(0, 0, 0, 12)12 - AIC:769.0817523205916
/usr/local/lib/python3.6/dist-packages/statsmodels/base/model.py:512: ConvergenceW
  "Check mle_retvals", ConvergenceWarning)
ARIMA(0, 0, 0)x(0, 0, 1, 12)12 - AIC:1526.6646435881128
ARIMA(0, 0, 0)x(0, 1, 0, 12)12 - AIC:477.71701309202774
ARIMA(0, 0, 0)x(1, 0, 0, 12)12 - AIC:497.2314433418337
/usr/local/lib/python3.6/dist-packages/statsmodels/base/model.py:512: ConvergenceW
  "Check mle_retvals", ConvergenceWarning)
ARIMA(0, 0, 0)x(1, 0, 1, 12)12 - AIC:1402.3936867800505
ARIMA(0, 0, 0)x(1, 1, 0, 12)12 - AIC:318.0047199116341
ARIMA(0, 0, 1)x(0, 0, 0, 12)12 - AIC:720.92522707581
ARIMA(0, 0, 1)x(0, 0, 1, 12)12 - AIC:2900.5279890044303
ARIMA(0, 0, 1)x(0, 1, 0, 12)12 - AIC:466.5607429809134
/usr/local/lib/python3.6/dist-packages/statsmodels/base/model.py:512: ConvergenceW
  "Check mle_retvals", ConvergenceWarning)
ARIMA(0, 0, 1)x(1, 0, 0, 12)12 - AIC:499.5730681144165
/usr/local/lib/python3.6/dist-packages/statsmodels/base/model.py:512: ConvergenceW
  "Check mle_retvals", ConvergenceWarning)
ARIMA(0, 0, 1)x(1, 0, 1, 12)12 - AIC:nan
ARIMA(0, 0, 1)x(1, 1, 0, 12)12 - AIC:319.98848769468657
ARIMA(0, 1, 0)x(0, 0, 0, 12)12 - AIC:677.894766843944
ARIMA(0, 1, 0)x(0, 0, 1, 12)12 - AIC:1420.8968510776567
ARIMA(0, 1, 0)x(0, 1, 0, 12)12 - AIC:486.6378567198382
ARIMA(0, 1, 0)x(1, 0, 0, 12)12 - AIC:497.78896630044073
ARIMA(0, 1, 0)x(1, 0, 1, 12)12 - AIC:1283.8048690707233
/usr/local/lib/python3.6/dist-packages/statsmodels/base/model.py:512: ConvergenceW
  "Check mle_retvals", ConvergenceWarning)
ARIMA(0, 1, 0)x(1, 1, 0, 12)12 - AIC:319.7714068109211
ARIMA(0, 1, 1)x(0, 0, 0, 12)12 - AIC:649.9056176817081
ARIMA(0, 1, 1)x(0, 0, 1, 12)12 - AIC:2568.9604692368553
ARIMA(0, 1, 1)x(0, 1, 0, 12)12 - AIC:458.8705548482836
ARIMA(0, 1, 1)x(1, 0, 0, 12)12 - AIC:486.1832977442527
ARIMA(0, 1, 1)x(1, 0, 1, 12)12 - AIC:2982.3212858304873
ARIMA(0, 1, 1)x(1, 1, 0, 12)12 - AIC:310.7574368417338
ARIMA(1, 0, 0)x(0, 0, 0, 12)12 - AIC:692.1645522067712
ARIMA(1, 0, 0)x(0, 0, 1, 12)12 - AIC:1378.8837536991703
ARIMA(1, 0, 0)x(0, 1, 0, 12)12 - AIC:479.46321478521355
/usr/local/lib/python3.6/dist-packages/statsmodels/base/model.py:512: ConvergenceW
  "Check mle_retvals", ConvergenceWarning)
ARIMA(1, 0, 0)x(1, 0, 0, 12)12 - AIC:480.9259367935194
ARIMA(1, 0, 0)x(1, 0, 1, 12)12 - AIC:1355.0241301172382
ARIMA(1, 0, 0)x(1, 1, 0, 12)12 - AIC:304.4664675084582
ARIMA(1, 0, 1)x(0, 0, 0, 12)12 - AIC:665.779444218597
ARIMA(1, 0, 1)x(0, 0, 1, 12)12 - AIC:3132.313062242019
ARIMA(1, 0, 1)x(0, 1, 0, 12)12 - AIC:468.36851958151317
ARIMA(1, 0, 1)x(1, 0, 0, 12)12 - AIC:482.5763323876961
ARIMA(1, 0, 1)x(1, 0, 1, 12)12 - AIC:2763.15984565311
ARIMA(1, 0, 1)x(1, 1, 0, 12)12 - AIC:306.01560021460404
ARIMA(1, 1, 0)x(0, 0, 0, 12)12 - AIC:671.2513547541902
ARIMA(1, 1, 0)x(0, 0, 1, 12)12 - AIC:1412.2716724812103
ARIMA(1, 1, 0)x(0, 1, 0, 12)12 - AIC:479.20034222811347
ARIMA(1, 1, 0)x(1, 0, 0, 12)12 - AIC:475.3403658784957
```

```
/usr/local/lib/python3.6/dist-packages/statsmodels/base/model.py:512: ConvergenceW
  "Check mle_retvals", ConvergenceWarning)
ARIMA(1, 1, 0)x(1, 0, 1, 12)12 - AIC:1284.3702695020268
ARIMA(1, 1, 0)x(1, 1, 0, 12)12 - AIC:300.6270901345411
ARIMA(1, 1, 1)x(0, 0, 0, 12)12 - AIC:649.0318019835028
ARIMA(1, 1, 1)x(0, 0, 1, 12)12 - AIC:2726.367540279938
```

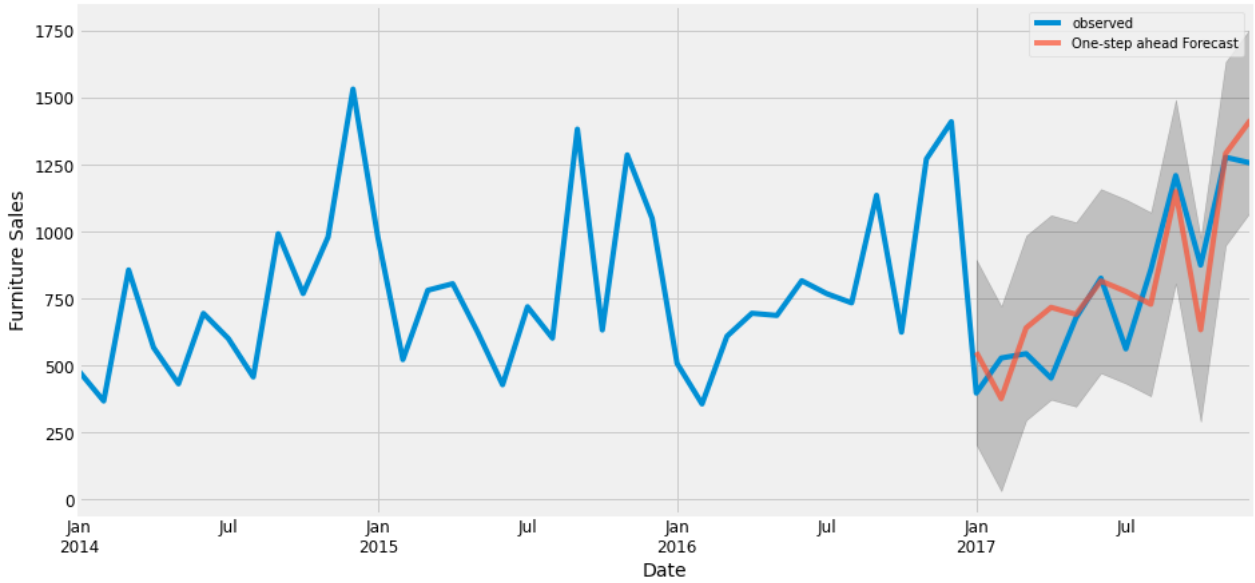
```
mod = sm.tsa.statespace.SARIMAX(y,
                                order=(1, 1, 1),
                                seasonal_order=(1, 1, 0, 12),
                                enforce_stationarity=False,
                                enforce_invertibility=False)
results = mod.fit()
print(results.summary().tables[1])
```

```
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.0146      0.342      0.043      0.966      -0.655       0.684
ma.L1         -1.0000      0.360     -2.781      0.005      -1.705      -0.295
ar.S.L12      -0.0253      0.042     -0.609      0.543      -0.107       0.056
sigma2      2.958e+04   1.22e-05   2.43e+09      0.000    2.96e+04    2.96e+04
==============================================================================
```

```
results.plot_diagnostics(figsize=(16, 8))
plt.show()
```



```
pred = results.get_prediction(start=pd.to_datetime('2017-01-01'), dynamic=False)
pred_ci = pred.conf_int()
ax = y['2014':].plot(label='observed')
pred.predicted_mean.plot(ax=ax, label='One-step ahead Forecast', alpha=.7, figsize=(14, 7)
ax.fill_between(pred_ci.index,
                pred_ci.iloc[:, 0],
                pred_ci.iloc[:, 1], color='k', alpha=.2)
ax.set_xlabel('Date')
ax.set_ylabel('Furniture Sales')
plt.legend()
plt.show()
```




```
y_forecasted = pred.predicted_mean
y_truth = y['2017-01-01':]
mse = ((y_forecasted - y_truth) ** 2).mean()
print('The Mean Squared Error of our forecasts is {}'.format(round(mse, 2)))
```

```
The Mean Squared Error of our forecasts is 22993.57
```

```
print('The Root Mean Squared Error of our forecasts is {}'.format(round(np.sqrt(mse), 2)))
```

```
The Root Mean Squared Error of our forecasts is 151.64
```

Double-click (or enter) to edit

Abhinay Bhatt - 179301009 - CSE A

To perform Text Analysis

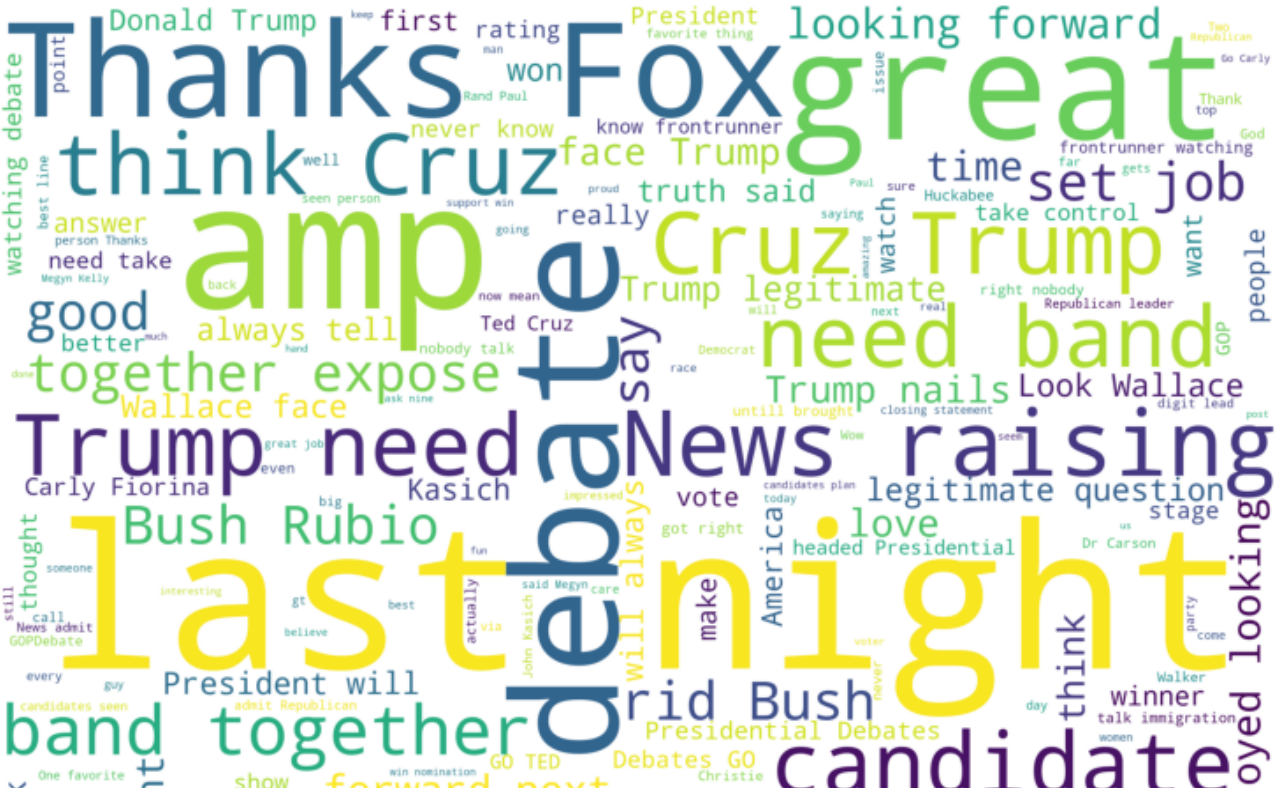
```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split

import nltk
from nltk.corpus import stopwords
from nltk.classify import SklearnClassifier

from wordcloud import WordCloud,STOPWORDS
import matplotlib.pyplot as plt
%matplotlib inline

from subprocess import check_output
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
True
```

```
data = pd.read_csv('/content/Sentiment.csv')
data = data[['text','sentiment']]


# Splitting the dataset into train and test set
train, test = train_test_split(data,test_size = 0.1)
# Removing neutral sentiments
train = train[train.sentiment != "Neutral"]


train_pos = train[ train['sentiment'] == 'Positive']
train_pos = train_pos['text']
train_neg = train[ train['sentiment'] == 'Negative']
train_neg = train_neg['text']

def wordcloud_draw(data, color = 'black'):
    words = ' '.join(data)
    cleaned_word = " ".join([word for word in words.split()
                            if 'http' not in word
                                and not word.startswith('@')
                                and not word.startswith('#')
                                and word != 'RT'
                            ])
    wordcloud = WordCloud(stopwords=STOPWORDS,
                      background_color=color,
                      width=2500,
```

```
                          height=2000
                         ).generate(cleaned_word)
        plt.figure(1,figsize=(13, 13))
        plt.imshow(wordcloud)
        plt.axis('off')
        plt.show()

    print("Positive words")
    wordcloud_draw(train_pos,'white')
    print("Negative words")
    wordcloud_draw(train_neg)
```

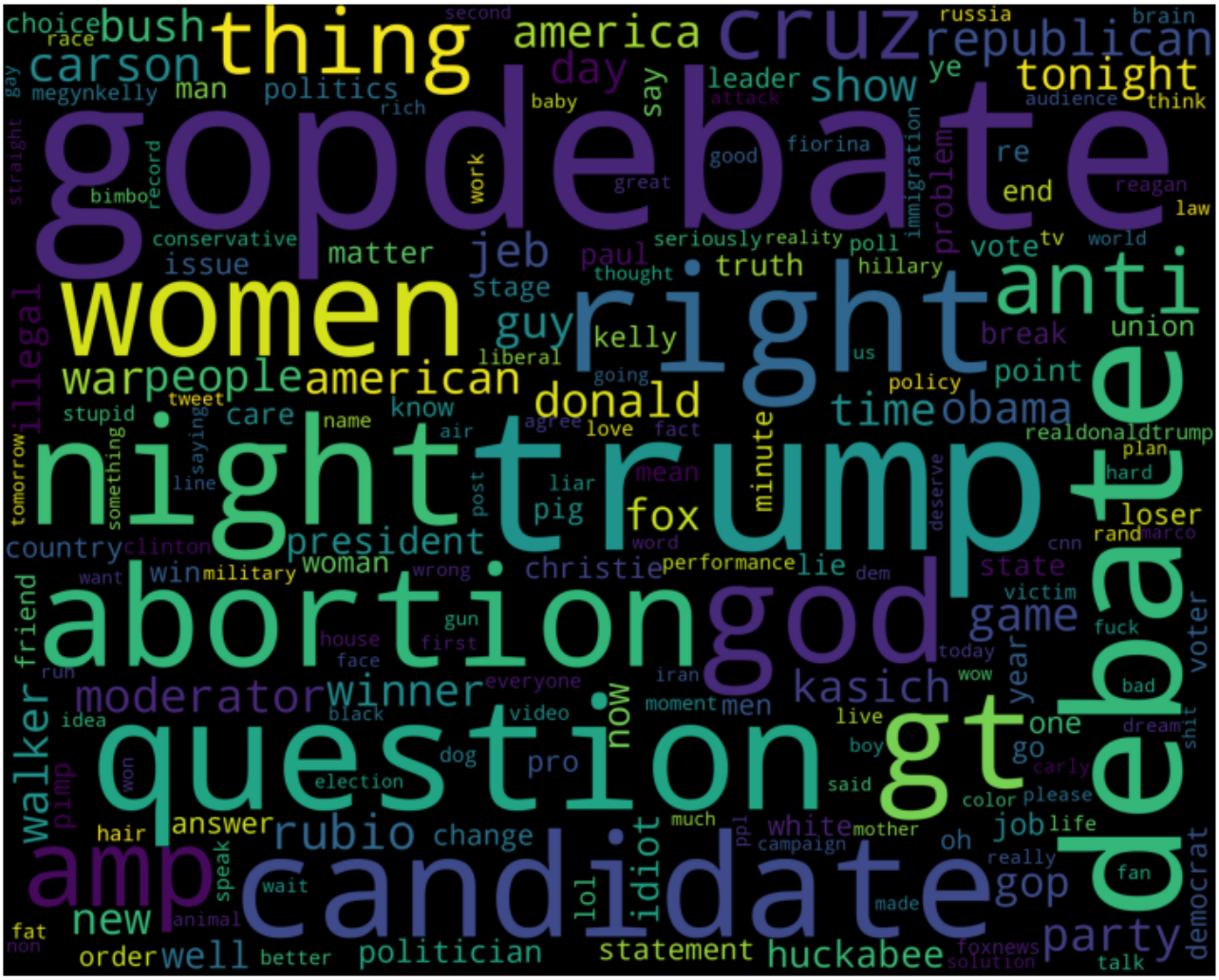
Positive words




```
tweets = []
stopwords_set = set(stopwords.words("english"))

for index, row in train.iterrows():
    words_filtered = [e.lower() for e in row.text.split() if len(e) >= 3]
    words_cleaned = [word for word in words_filtered
        if 'http' not in word
        and not word.startswith('@')
        and not word.startswith('#')
        and word != 'RT']
    words_without_stopwords = [word for word in words_cleaned if not word in stopwords_set
    tweets.append((words_without_stopwords, row.sentiment))

test_pos = test[ test['sentiment'] == 'Positive']
test_pos = test_pos['text']
test_neg = test[ test['sentiment'] == 'Negative']
test_neg = test_neg['text']
```


```
# Extracting word features
def get_words_in_tweets(tweets):
    all = []
    for (words, sentiment) in tweets:
        all.extend(words)
    return all

def get_word_features(wordlist):
    wordlist = nltk.FreqDist(wordlist)
    features = wordlist.keys()
    return features
w_features = get_word_features(get_words_in_tweets(tweets))
```

```
def extract_features(document):
    document_words = set(document)
    features = {}
    for word in w_features:
        features['contains(%s)' % word] = (word in document_words)
    return features


wordcloud_draw(w_features)
```




```
# Training the Naive Bayes classifier
training_set = nltk.classify.apply_features(extract_features,tweets)
classifier = nltk.NaiveBayesClassifier.train(training_set)



neg_cnt = 0
pos_cnt = 0
for obj in test_neg:
    res =  classifier.classify(extract_features(obj.split()))
    if(res == 'Negative'):
        neg_cnt = neg_cnt + 1
for obj in test_pos:
```

```
for obj in test_pos:
    res =  classifier.classify(extract_features(obj.split()))
    if(res == 'Positive'):
        pos_cnt = pos_cnt + 1

print('Negative: %s/%s '  % (len(test_neg),neg_cnt))
print('Positive: %s/%s '  % (len(test_pos),pos_cnt))
```

Negative: 817/789

Positive: 216/77