Name : Chetan Kamatagi
College : KLE TECHNOLOGICAL UNIVERSITY, HUBBALLI, KARNATAKA
Branch : Automations and Robotics
Year : 2nd year

Github Link: https://github.com/0311Chetan/MAJOR-PROJECT

MAJOR PROJECT 1:

```
# Major Project 1:
# Choose any dataset of your choice and apply a suitable CLASSIFIER/REGRESSOR.
```

```
# importing Required libraries
import pandas as pd
import numpy as np
df = pd.read_csv('https://raw.githubusercontent.com/ameenmanna8824/DATASETS/main/heart_dis
df
```

	Unnamed: 0	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	s
0	0	63	1	3	145	233	1	0	150	0	2.3	
1	1	37	1	2	130	250	0	1	187	0	3.5	
2	2	41	0	1	130	204	0	0	172	0	1.4	
3	3	56	1	1	120	236	0	1	178	0	0.8	
4	4	57	0	0	120	354	0	1	163	1	0.6	
...	...	...	...	...	...	...	...	...	...	...	...	
298	298	57	0	0	140	241	0	1	123	1	0.2	
299	299	45	1	3	110	264	0	1	132	0	1.2	
300	300	68	1	0	144	193	1	1	141	0	3.4	
301	301	57	1	0	130	131	0	1	115	1	1.2	
302	302	57	0	1	130	236	0	0	174	0	0.0	

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 193 entries, 0 to 192
Data columns (total 25 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   symboling          193 non-null    int64  
 1   make               193 non-null    object 
 2   fuel_type          193 non-null    object 
 3   aspiration         193 non-null    object 
 4   num_of_doors       193 non-null    int64  
 5   body_style         193 non-null    object 
 6   drive_wheels       193 non-null    object 
 7   engine_location    193 non-null    object 
 8   wheel_base         193 non-null    float64
 9   length             193 non-null    float64
 10  width              193 non-null    float64
 11  height             193 non-null    float64
 12  curb_weight        193 non-null    int64
 13  engine_type        193 non-null    object
 14  num_of_cylinders   193 non-null    int64
 15  engine_size        193 non-null    int64
 16  fuel_system        193 non-null    object
 17  bore               193 non-null    float64
 18  stroke             193 non-null    float64
 19  compression_ratio  193 non-null    int64
 20  horsepower         193 non-null    int64
 21  peak_rpm           193 non-null    int64
 22  city_mpg           193 non-null    int64
 23  highway_mpg        193 non-null    int64
 24  price              193 non-null    int64
dtypes: float64(6), int64(11), object(8)
memory usage: 37.8+ KB
```

```
# Preprocessing of the data is not required
# because dataset is complete and there are no null values.
```

```
# Dividing the data into Input and Output
x = df.iloc[:,4:9].values
x
```

```
array([[145, 233,   1,   0, 150],
       [130, 250,   0,   1, 187],
       [130, 204,   0,   0, 172],
       ...,
       [144, 193,   1,   1, 141],
       [130, 131,   0,   1, 115],
       [130, 236,   0,   0, 174]])
```

```
y = df.iloc[:,14:15].values
y
```

```
# Train and Test variables
from sklearn.model_selection import train_test_split
xtr,xts,ytr,yts = train_test_split(x,y,random_state = 0)
```

```
print(x.shape)
print(xtr.shape)
print(xts.shape)

print(y.shape)
print(ytr.shape)
print(yts.shape)
```

```
(303, 5)
(227, 5)
(76, 5)
(303, 1)
(227, 1)
(76, 1)
```

```
# Normalizing is not necessary since the output and input are already scaled
```

```
# Running a Classifier/Regressor
from sklearn.linear_model import LogisticRegression
mdl = LogisticRegression()
```

```
# Fitting the model to the classifier:
mdl.fit(xtr,ytr)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:993: DataConversi
  y = column_or_1d(y, warn=True)
LogisticRegression()
```

```
# predicting the output:
ypred=mdl.predict(xts)
ypred
```

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

```
# Accuracy:
from sklearn.metrics import accuracy_score
accuracy_score(ypred,yts) *100
```

```
76.31578947368422
```

```
# Predicting the individual values
mdl.predict([[180,190,1,0,200]])
```

```
array([1])
```

```
mdl.predict([[80,90,1,0,20]])
```

```
array([0])
```

MAJOR PROJECT 2:

```
# Major project 2:
# Choose any dataset of your choice and apply K Means Clustering.
```

```
# Import the Required Libraries

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

```
# Importing the dataset from the github

df = pd.read_csv('https://raw.githubusercontent.com/0311Chetan/Codes/main/concrete.csv')
df
```

	Cement	BlastFurnaceSlag	FlyAsh	Water	Superplasticizer	CoarseAggregate	Fi
0	540.0	0.0	0.0	162.0	2.5	1040.0	
1	540.0	0.0	0.0	162.0	2.5	1055.0	
2	332.5	142.5	0.0	228.0	0.0	932.0	
3	332.5	142.5	0.0	228.0	0.0	932.0	
4	198.6	132.4	0.0	192.0	0.0	978.4	
...	...	...	...	...	...	...	
1025	276.4	116.0	90.3	179.6	8.9	870.1	
1026	322.2	0.0	115.6	196.0	10.4	817.9	
1027	148.5	139.4	108.6	192.7	6.1	892.4	
1028	159.1	186.7	0.0	175.6	11.3	989.6	
1029	260.9	100.5	78.3	200.6	8.6	864.5	

1030 rows × 9 columns

```
df.shape
```

```
(1030, 9)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1030 entries, 0 to 1029
Data columns (total 9 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   Cement               1030 non-null   float64
 1   BlastFurnaceSlag     1030 non-null   float64
 2   FlyAsh               1030 non-null   float64
 3   Water                1030 non-null   float64
 4   Superplasticizer     1030 non-null   float64
 5   CoarseAggregate      1030 non-null   float64
 6   FineAggregate        1030 non-null   float64
 7   Age                  1030 non-null   int64  
 8   CompressiveStrength  1030 non-null   float64
dtypes: float64(8), int64(1)
memory usage: 72.5 KB
```

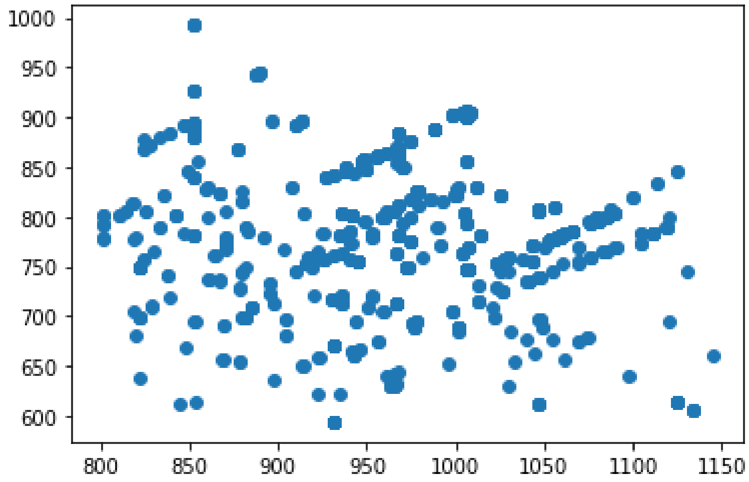
```
# Here we take input parameters as :CoarseAggregate and FineAggregate  :
```

```
# Now dividing the data into Input and output:
x = df.iloc[:,5:7].values
x
```

```
array([[1040. ,  676. ],
       [1055. ,  676. ],
       [ 932. ,  594. ],
       ...,
       [ 892.4,  780. ],
       [ 989.6,  788.9],
       [ 864.5,  761.5]])
```

```
#VISUALISATION of the input data before applying any clustering techiques :

plt.scatter(df['CoarseAggregate'],df['FineAggregate'])
```

```
<matplotlib.collections.PathCollection at 0x7fcff987b050>
```



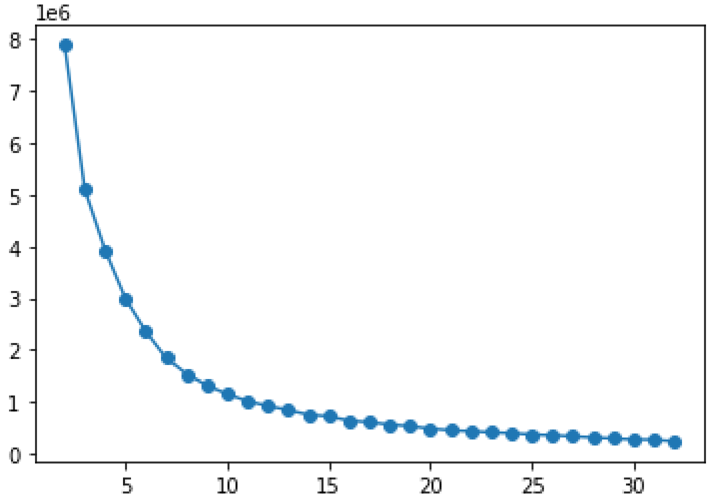
```
#Now Finding out the number of clusters(k)
# Total number of points = 1030

np.sqrt(1030)

# k value ranges from 2 - 32
```

```
32.09361307176243
```

```
# Now we have to find No. of clusters(k)
```

```
#1.ELBOW METHOD
from sklearn.cluster import KMeans
k = range(2,33)

sse = []

for i in k :
  model_demo = KMeans(n_clusters = i,random_state = 0)
  model_demo.fit(x)
  sse.append(model_demo.inertia_)
plt.scatter(k,sse)
plt.plot(k,sse)
```

```
    [<matplotlib.lines.Line2D at 0x7fcff7f514d0>]
```



```
# 2.SILHOUETTE SCORE METHOD

from sklearn.metrics import silhouette_score
k = range(2,33)
for i in k:
  model_demo = KMeans(n_clusters = i,random_state = 0)
  model_demo.fit(x)
  y_pred = model_demo.predict(x)
  print(f"{i} Clusters, Score = {silhouette_score(x,y_pred)}")
  plt.bar(i,silhouette_score(x,y_pred))
```

```
     2 Clusters, Score = 0.3558185095888853
     3 Clusters, Score = 0.409974801365556
     4 Clusters, Score = 0.4130838519874459
     5 Clusters, Score = 0.4486948545137604
     6 Clusters, Score = 0.45199448215467036
     7 Clusters, Score = 0.470588180756573
     8 Clusters, Score = 0.46567284221181254
     9 Clusters, Score = 0.4636699148596504
     10 Clusters, Score = 0.4691868710743738
     11 Clusters, Score = 0.46267961260749596
     12 Clusters, Score = 0.4658701556028496
     13 Clusters, Score = 0.4885732054608967
     14 Clusters, Score = 0.4828828733854631
     15 Clusters, Score = 0.46943457773884695
     16 Clusters, Score = 0.4796691111373154
     17 Clusters, Score = 0.4840973594478049
     18 Clusters, Score = 0.47860901217317503
     19 Clusters, Score = 0.48117825012833304
     20 Clusters, Score = 0.48395282373636606
     21 Clusters, Score = 0.4972722120915127
     22 Clusters, Score = 0.4847935953371792
     23 Clusters, Score = 0.5018159496160546
```

```
# From the SILHOUETTE SCORE METHOD we can clearly observe that there can be 32 clusters ca
# THE No OF CLUSTERS TO BE CONSIDERED IS 32.
```

```
     28 Clusters, Score = 0.5280081121806302
```

```
# 7.APPLY CLUSTERER
k = 32
from sklearn.cluster import KMeans

model = KMeans(n_clusters = k,random_state = 0)
model.fit(x)
```

```
     KMeans(n_clusters=32, random_state=0)
```

```
y = model.predict(x) # predicted output
y
```

```
     array([31, 31, 10, ...,  1,  2,  1], dtype=int32)
```

```
y.size
```

```
     1030
```

```
x[y == 1,1]
#so the first '1' is cluster no 1 and the second '1' is column index 1
#the value of input,when cluster 1 is selected and column index 1 selected
```

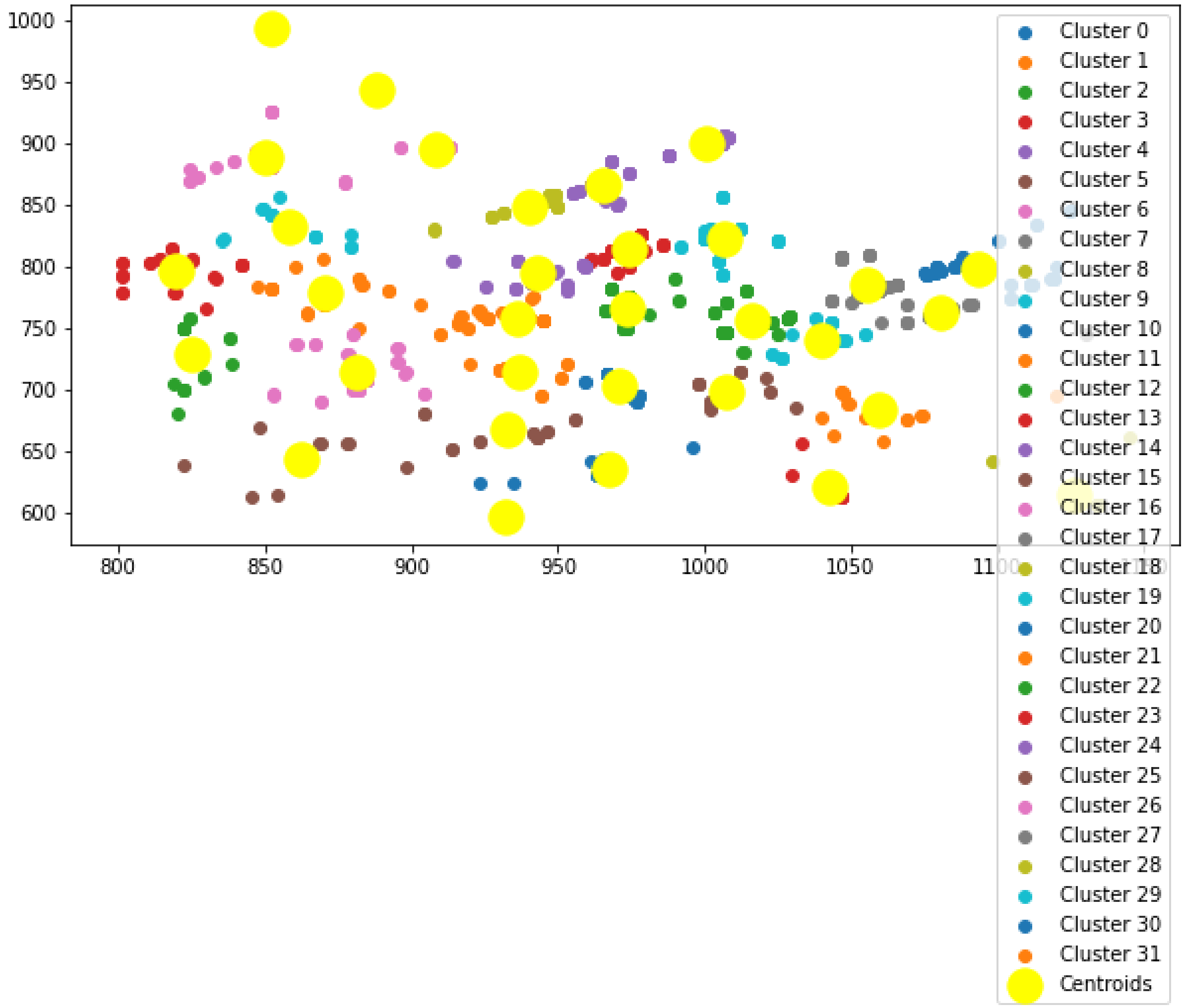
```
     array([781.5, 781.5, 781.5, 781.5, 781.5, 805. , 768. , 800. , 750. ,
            785. , 774. , 790. , 780. , 768. , 780. , 761. , 783. , 785.3,
            774. , 790. , 779.7, 768.3, 780. , 761.5])
```

```
np.unique(y,return_counts = True)
```

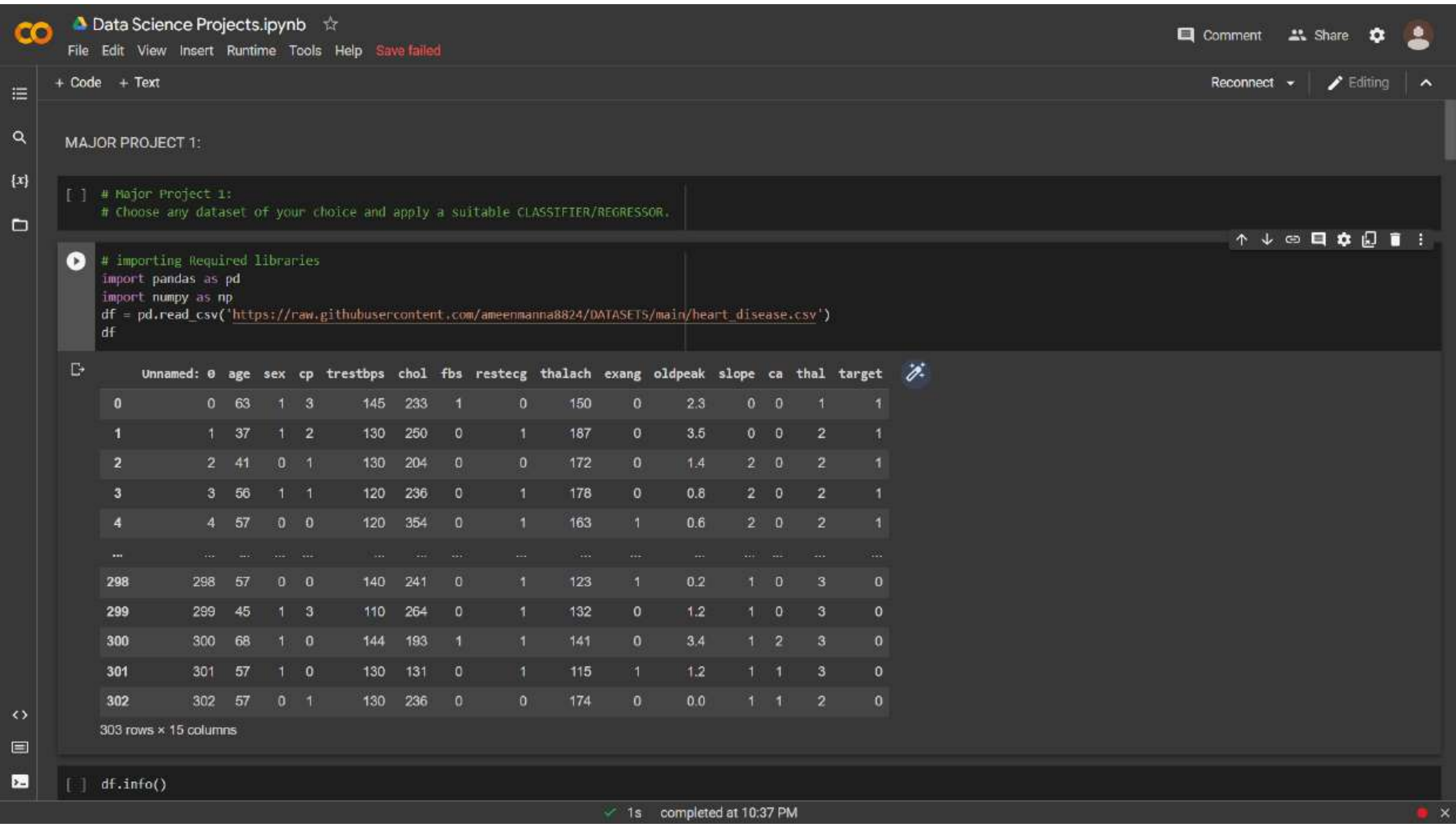
```
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
        17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31],
       dtype=int32),
 array([68, 24, 35,  7, 51, 39, 44, 12, 31, 31, 32, 56, 20, 37, 40, 20, 31,
        87, 52, 21, 15, 22, 64, 23, 43,  9, 10, 21,  5, 37, 28, 15]))
```

```
#FINAL VISUALISATION
plt.figure(figsize = (10,5))
for i in range(k):
  plt.scatter(x[y == i,0],x[y == i,1],label = f'Cluster {i}')
plt.scatter(model.cluster_centers_[:,0],model.cluster_centers_[:,1],s = 300,c = 'yellow',
            label = 'Centroids')
plt.legend()
```

```
<matplotlib.legend.Legend at 0x7fcff7b975d0>
```



✓ 1s completed at 10:37 PM ● ×



Data Science Projects.ipynb
File Edit View Insert Runtime Tools Help Save failed
Comment
Share
+ Code + Text
Reconnect
Editing

MAJOR PROJECT 1:

```
[ ] # Major Project 1:
    # Choose any dataset of your choice and apply a suitable CLASSIFIER/REGRESSOR.
```

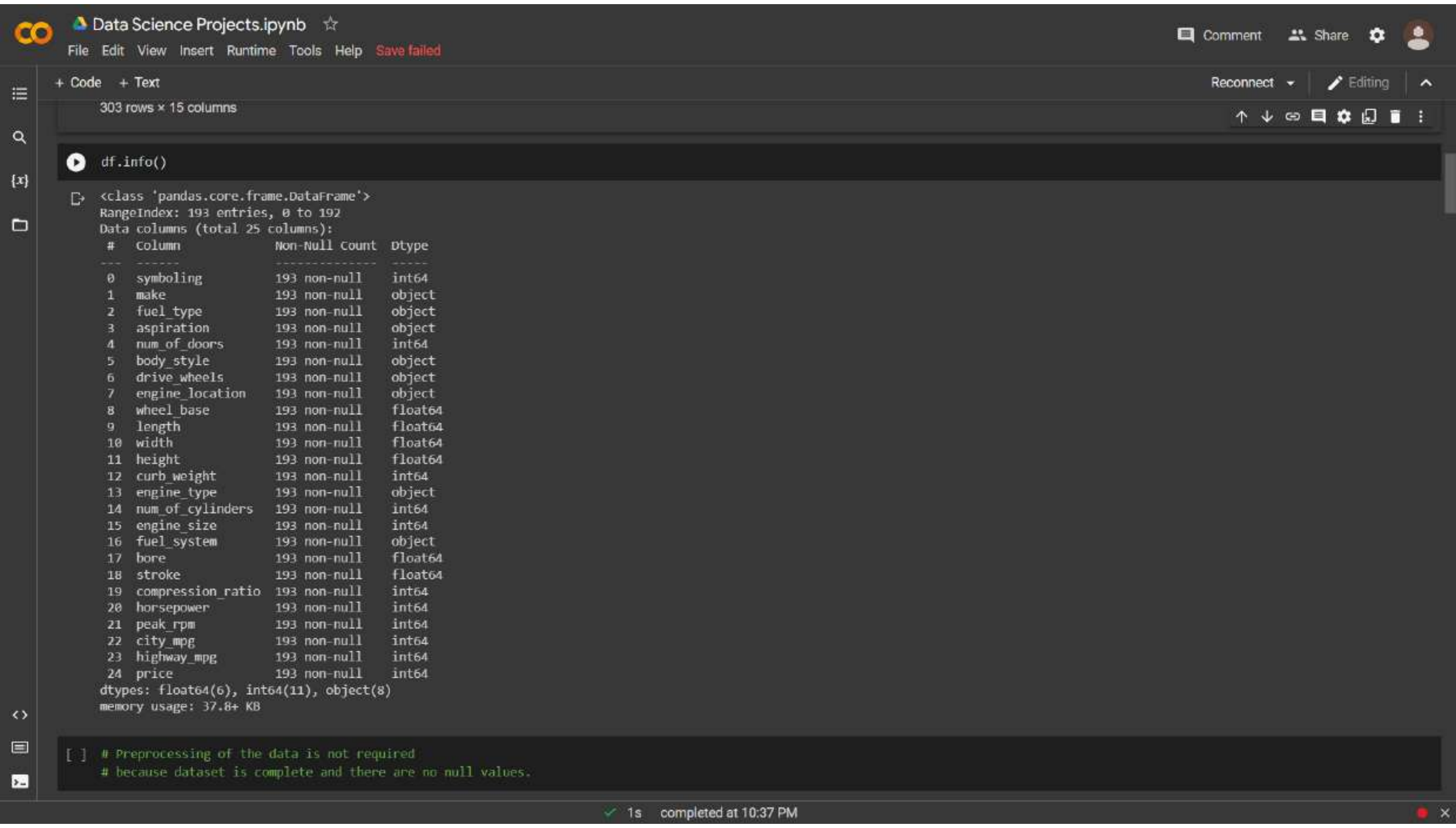
```
# importing Required libraries
import pandas as pd
import numpy as np
df = pd.read_csv('https://raw.githubusercontent.com/ameenmanna8824/DATASETS/main/heart_disease.csv')
df
```

	Unnamed: 0	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
298	298	57	0	0	140	241	0	1	123	1	0.2	1	0	3	0
299	299	45	1	3	110	264	0	1	132	0	1.2	1	0	3	0
300	300	68	1	0	144	193	1	1	141	0	3.4	1	2	3	0
301	301	57	1	0	130	131	0	1	115	1	1.2	1	1	3	0
302	302	57	0	1	130	236	0	0	174	0	0.0	1	1	2	0

303 rows × 15 columns

```
[ ] df.info()
```

✓ 1s completed at 10:37 PM

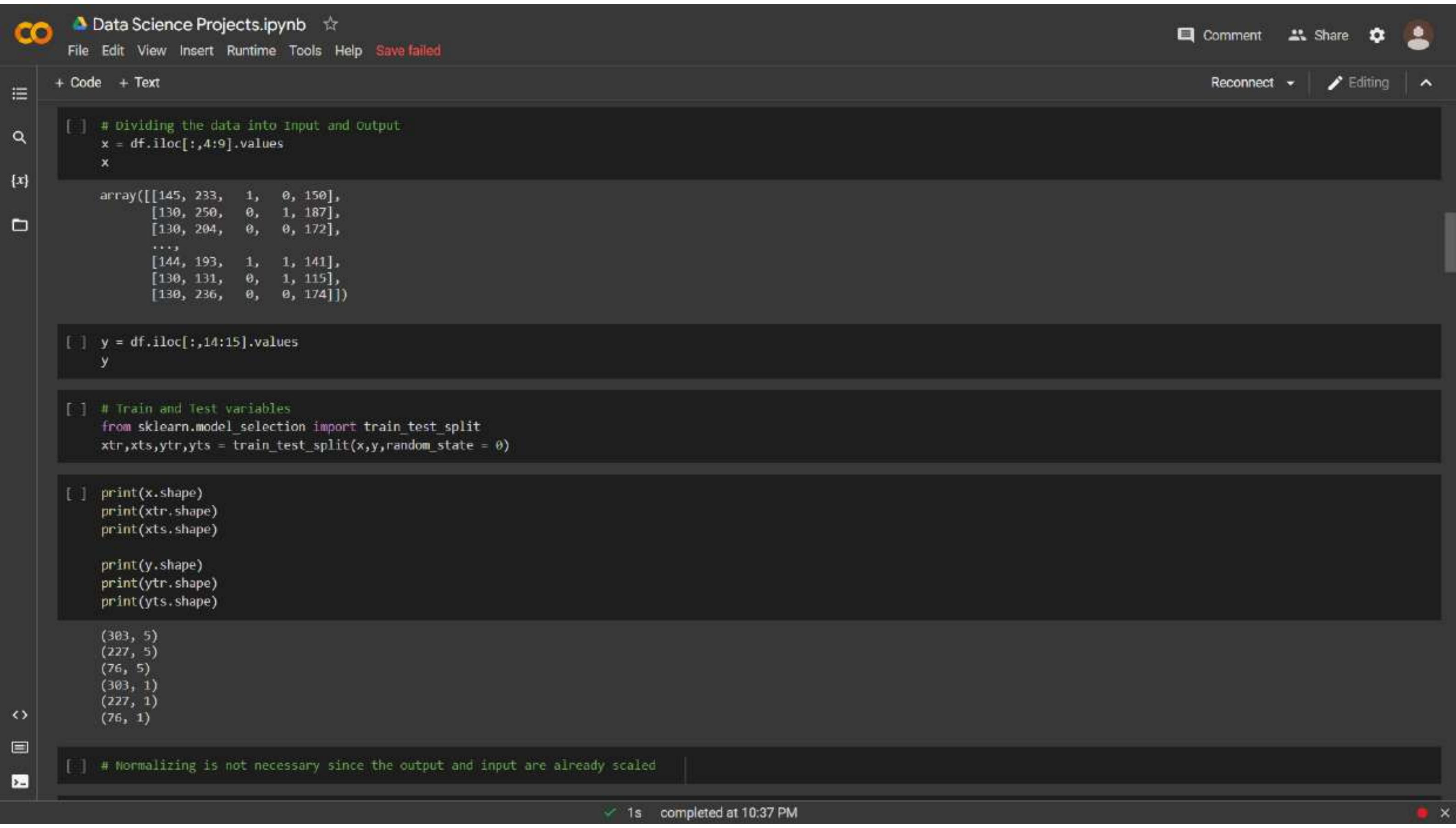

Data Science Projects.ipynb
File Edit View Insert Runtime Tools Help Save failed
Comment
Share
+ Code + Text
Reconnect
Editing
1s completed at 10:37 PM

303 rows × 15 columns

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 193 entries, 0 to 192
Data columns (total 25 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   symboling          193 non-null    int64
 1   make               193 non-null    object
 2   fuel_type          193 non-null    object
 3   aspiration         193 non-null    object
 4   num_of_doors       193 non-null    int64
 5   body_style         193 non-null    object
 6   drive_wheels       193 non-null    object
 7   engine_location    193 non-null    object
 8   wheel_base         193 non-null    float64
 9   length             193 non-null    float64
 10  width              193 non-null    float64
 11  height             193 non-null    float64
 12  curb_weight        193 non-null    int64
 13  engine_type        193 non-null    object
 14  num_of_cylinders   193 non-null    int64
 15  engine_size        193 non-null    int64
 16  fuel_system        193 non-null    object
 17  bore               193 non-null    float64
 18  stroke             193 non-null    float64
 19  compression_ratio  193 non-null    int64
 20  horsepower         193 non-null    int64
 21  peak_rpm           193 non-null    int64
 22  city_mpg           193 non-null    int64
 23  highway_mpg        193 non-null    int64
 24  price              193 non-null    int64
dtypes: float64(6), int64(11), object(8)
memory usage: 37.8+ KB
```

```
[ ] # Preprocessing of the data is not required
    # because dataset is complete and there are no null values.
```


Data Science Projects.ipynb
File Edit View Insert Runtime Tools Help Save failed
Comment Share
+ Code + Text
Reconnect
Editing
1s completed at 10:37 PM

```
# Dividing the data into Input and Output
x = df.iloc[:,4:9].values
x
```

```
array([[145, 233,   1,   0, 150],
       [130, 250,   0,   1, 187],
       [130, 204,   0,   0, 172],
       ...,
       [144, 193,   1,   1, 141],
       [130, 131,   0,   1, 115],
       [130, 236,   0,   0, 174]])
```

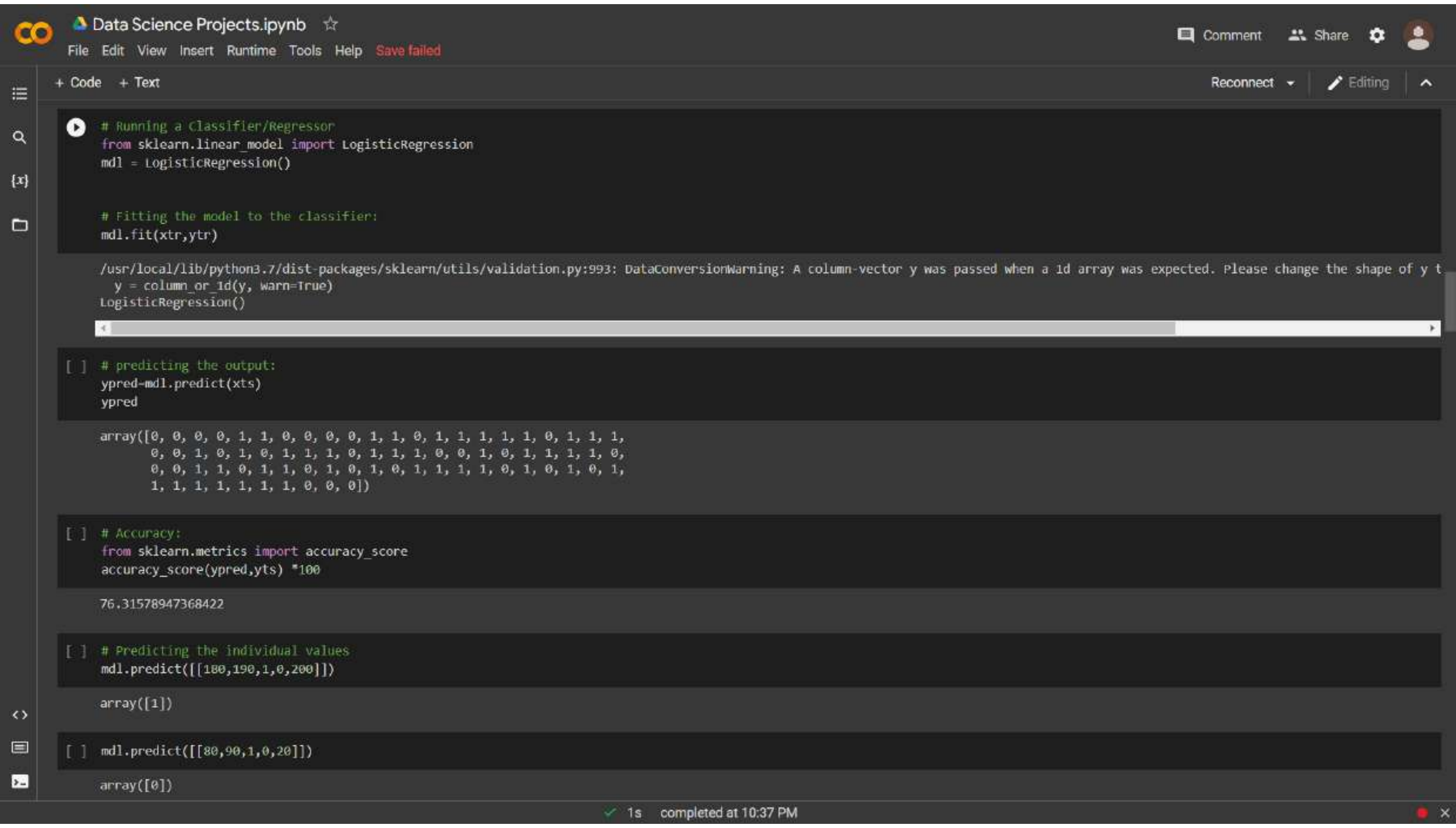
```
y = df.iloc[:,14:15].values
y
```

```
# Train and Test variables
from sklearn.model_selection import train_test_split
xtr,xts,ytr,yts = train_test_split(x,y,random_state = 0)
```

```
print(x.shape)
print(xtr.shape)
print(xts.shape)

print(y.shape)
print(ytr.shape)
print(yts.shape)
```

```
(303, 5)
(227, 5)
(76, 5)
(303, 1)
(227, 1)
(76, 1)
```

```
# Normalizing is not necessary since the output and input are already scaled
```


Data Science Projects.ipynb
File Edit View Insert Runtime Tools Help Save failed
Comment
Share
+ Code + Text
Reconnect
Editing
✓ 1s completed at 10:37 PM

```
# Running a Classifier/Regressor
from sklearn.linear_model import LogisticRegression
mdl = LogisticRegression()


# Fitting the model to the classifier:
mdl.fit(xtr,ytr)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:993: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y t
  y = column_or_1d(y, warn=True)
LogisticRegression()
```

```
[ ] # predicting the output:
    ypred-mdl.predict(xts)
    ypred
```

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

```
[ ] # Accuracy:
    from sklearn.metrics import accuracy_score
    accuracy_score(ypred,yts) *100
```

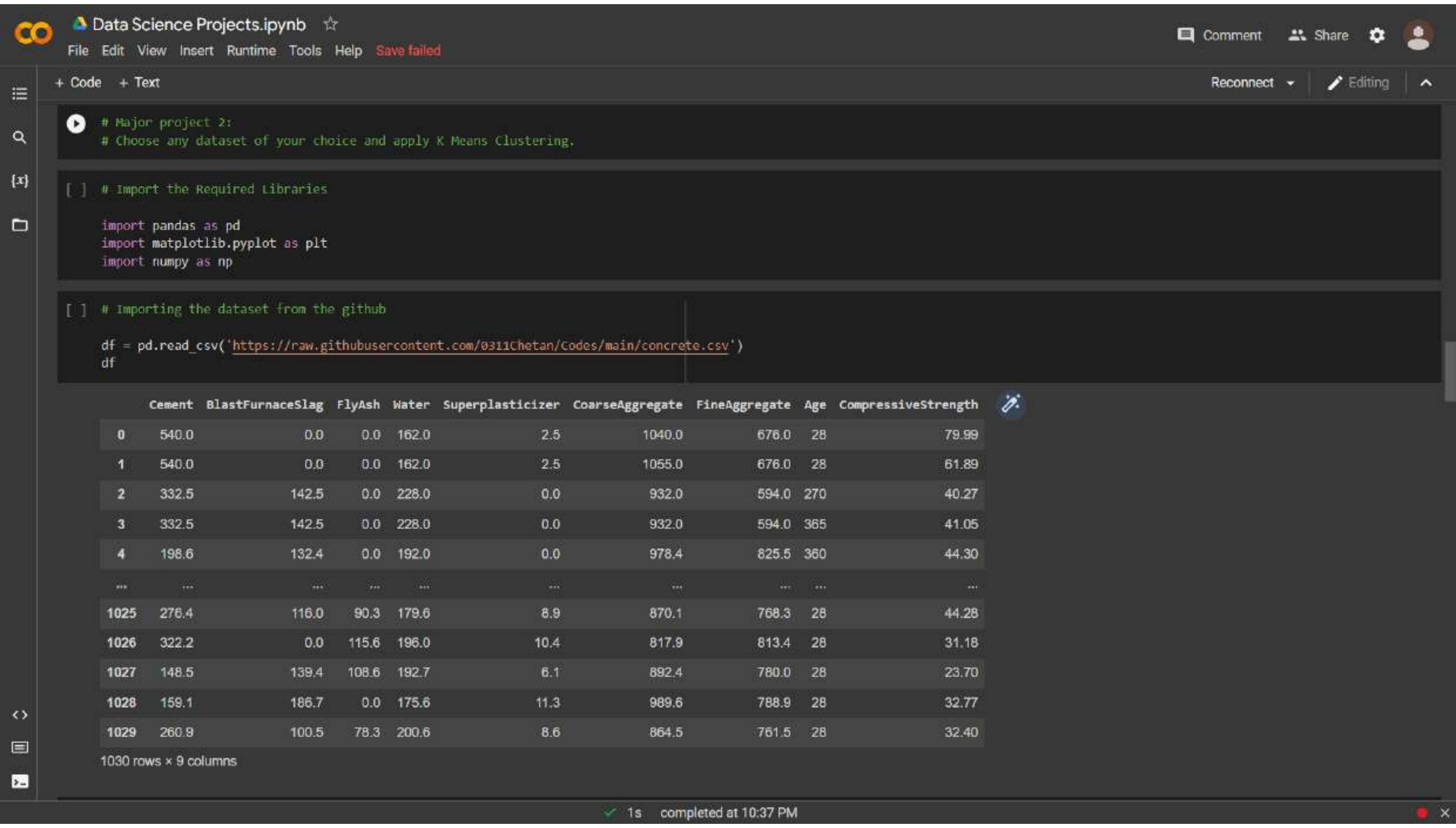
```
76.31578947368422
```

```
[ ] # Predicting the individual values
    mdl.predict([[180,190,1,0,200]])
```

```
array([1])
```

```
[ ] mdl.predict([[80,90,1,0,20]])
```

```
array([0])
```


Data Science Projects.ipynb
File Edit View Insert Runtime Tools Help Save failed
Comment Share
+ Code + Text
Reconnect Editing

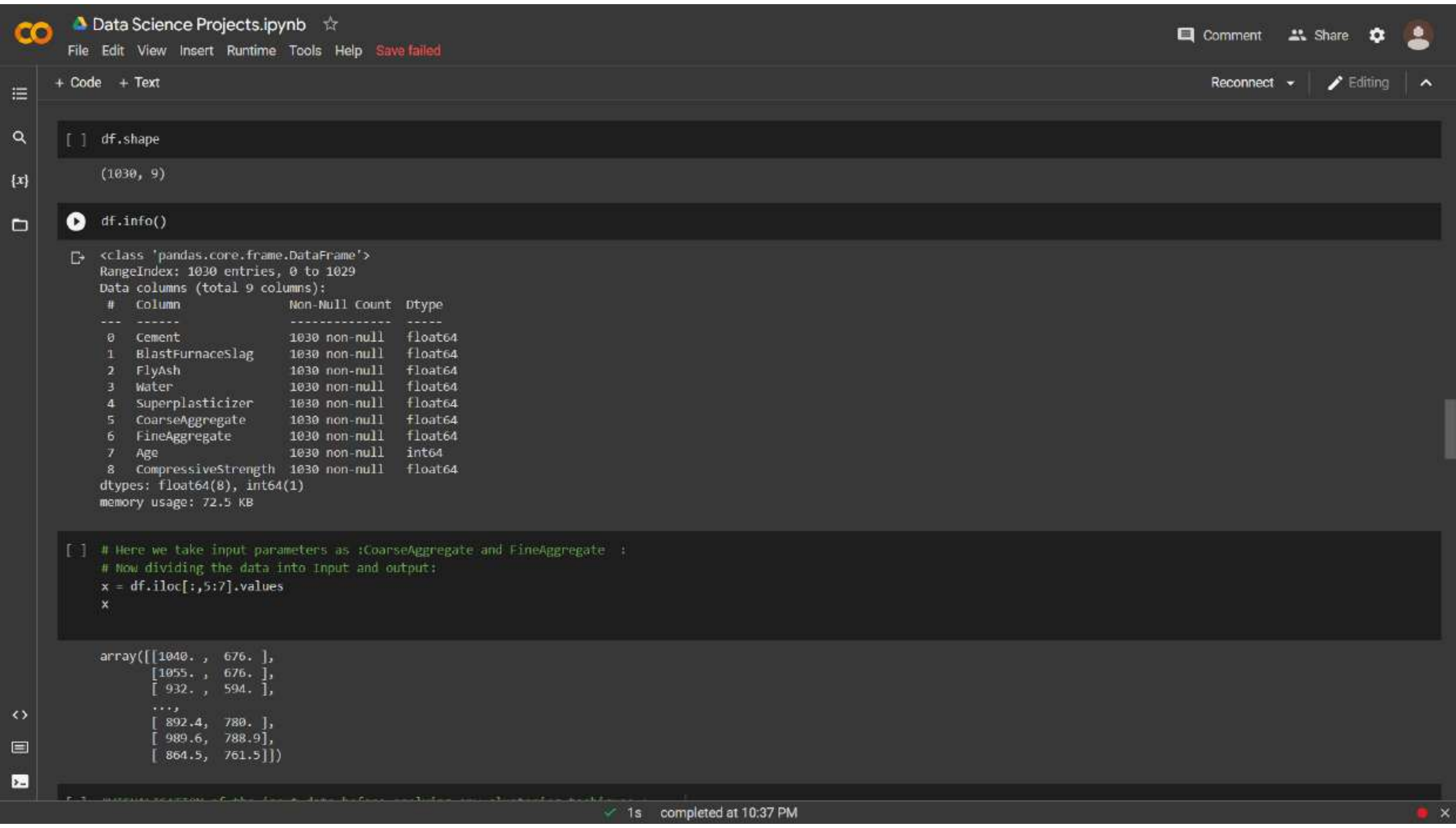
```
# Major project 2:
# Choose any dataset of your choice and apply K Means Clustering.
```

```
# Import the Required Libraries

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

```
# Importing the dataset from the github

df = pd.read_csv('https://raw.githubusercontent.com/0311Chetan/Codes/main/concrete.csv')
df
```

	Cement	BlastFurnaceSlag	FlyAsh	Water	Superplasticizer	CoarseAggregate	FineAggregate	Age	CompressiveStrength
0	540.0	0.0	0.0	162.0	2.5	1040.0	676.0	28	79.99
1	540.0	0.0	0.0	162.0	2.5	1055.0	676.0	28	61.89
2	332.5	142.5	0.0	228.0	0.0	932.0	594.0	270	40.27
3	332.5	142.5	0.0	228.0	0.0	932.0	594.0	365	41.05
4	198.6	132.4	0.0	192.0	0.0	978.4	825.5	360	44.30
...	...	...	...	...	...	...	...	...	...
1025	276.4	116.0	90.3	179.6	8.9	870.1	768.3	28	44.28
1026	322.2	0.0	115.6	196.0	10.4	817.9	813.4	28	31.18
1027	148.5	139.4	108.6	192.7	6.1	892.4	780.0	28	23.70
1028	159.1	186.7	0.0	175.6	11.3	989.6	788.9	28	32.77
1029	260.9	100.5	78.3	200.6	8.6	864.5	761.5	28	32.40

1030 rows × 9 columns


Data Science Projects.ipynb
File Edit View Insert Runtime Tools Help Save failed
Comment Share
+ Code + Text
Reconnect
Editing
✓ 1s completed at 10:37 PM

```
df.shape
```

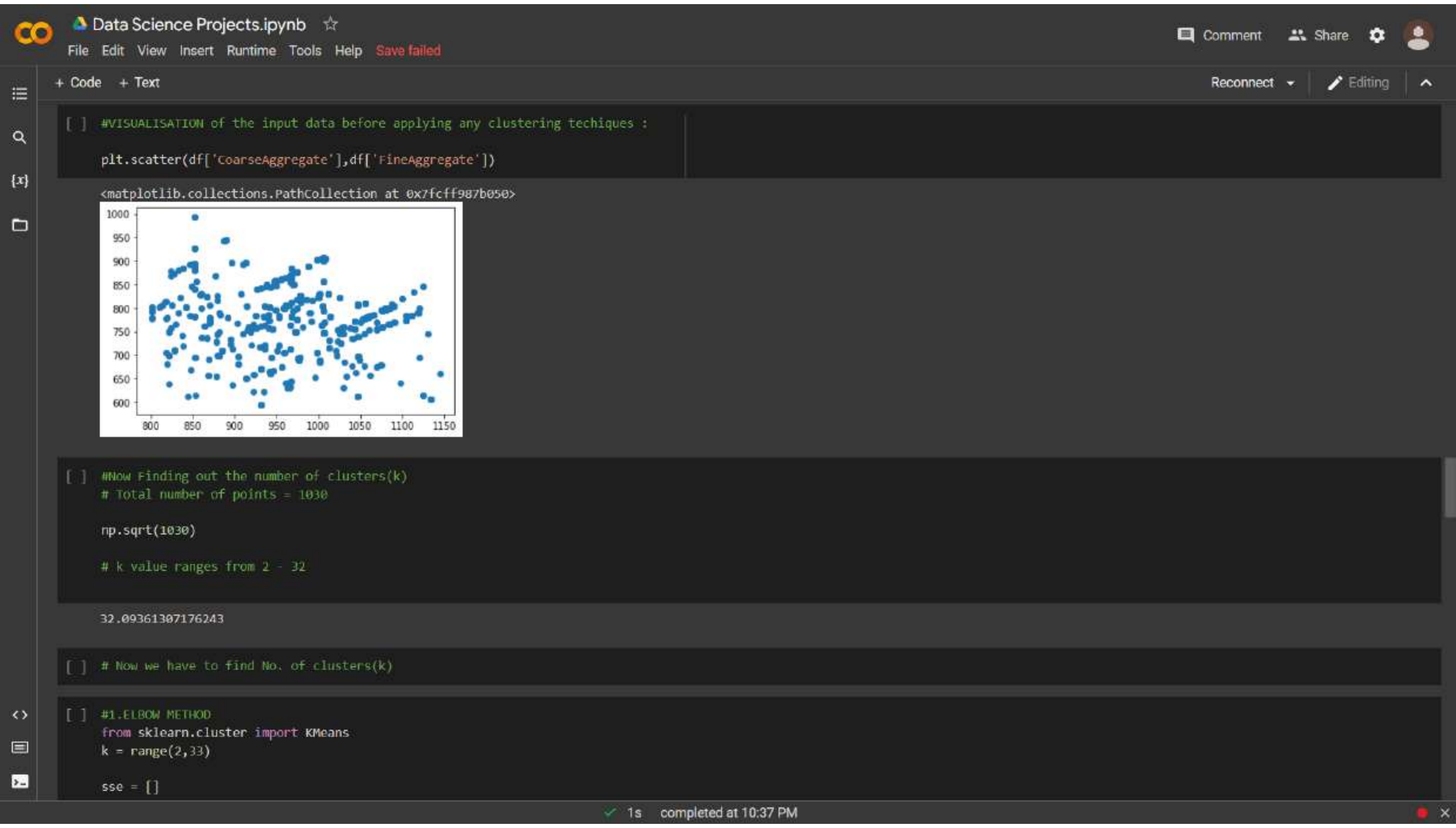
```
(1030, 9)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1030 entries, 0 to 1029
Data columns (total 9 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   Cement               1030 non-null   float64
 1   BlastFurnaceSlag     1030 non-null   float64
 2   FlyAsh               1030 non-null   float64
 3   Water                1030 non-null   float64
 4   Superplasticizer     1030 non-null   float64
 5   CoarseAggregate      1030 non-null   float64
 6   FineAggregate        1030 non-null   float64
 7   Age                  1030 non-null   int64  
 8   CompressiveStrength  1030 non-null   float64
dtypes: float64(8), int64(1)
memory usage: 72.5 KB
```

```
# Here we take input parameters as :CoarseAggregate and FineAggregate  :
# Now dividing the data into Input and output:
x = df.iloc[:,5:7].values
x
```

```
array([[1040. ,  676. ],
       [1055. ,  676. ],
       [ 932. ,  594. ],
       ...,
       [ 892.4,  780. ],
       [ 989.6,  788.9],
       [ 864.5,  761.5]])
```


Data Science Projects.ipynb
File Edit View Insert Runtime Tools Help Save failed
Comment
Share
+ Code + Text
Reconnect
Editing
1000
950
900
850
800
750
700
650
600
800
850
900
950
1000
1050
1100
1150
✓ 1s completed at 10:37 PM

```
[ ] #VISUALISATION of the input data before applying any clustering techiques :

    plt.scatter(df['CoarseAggregate'],df['FineAggregate'])
```

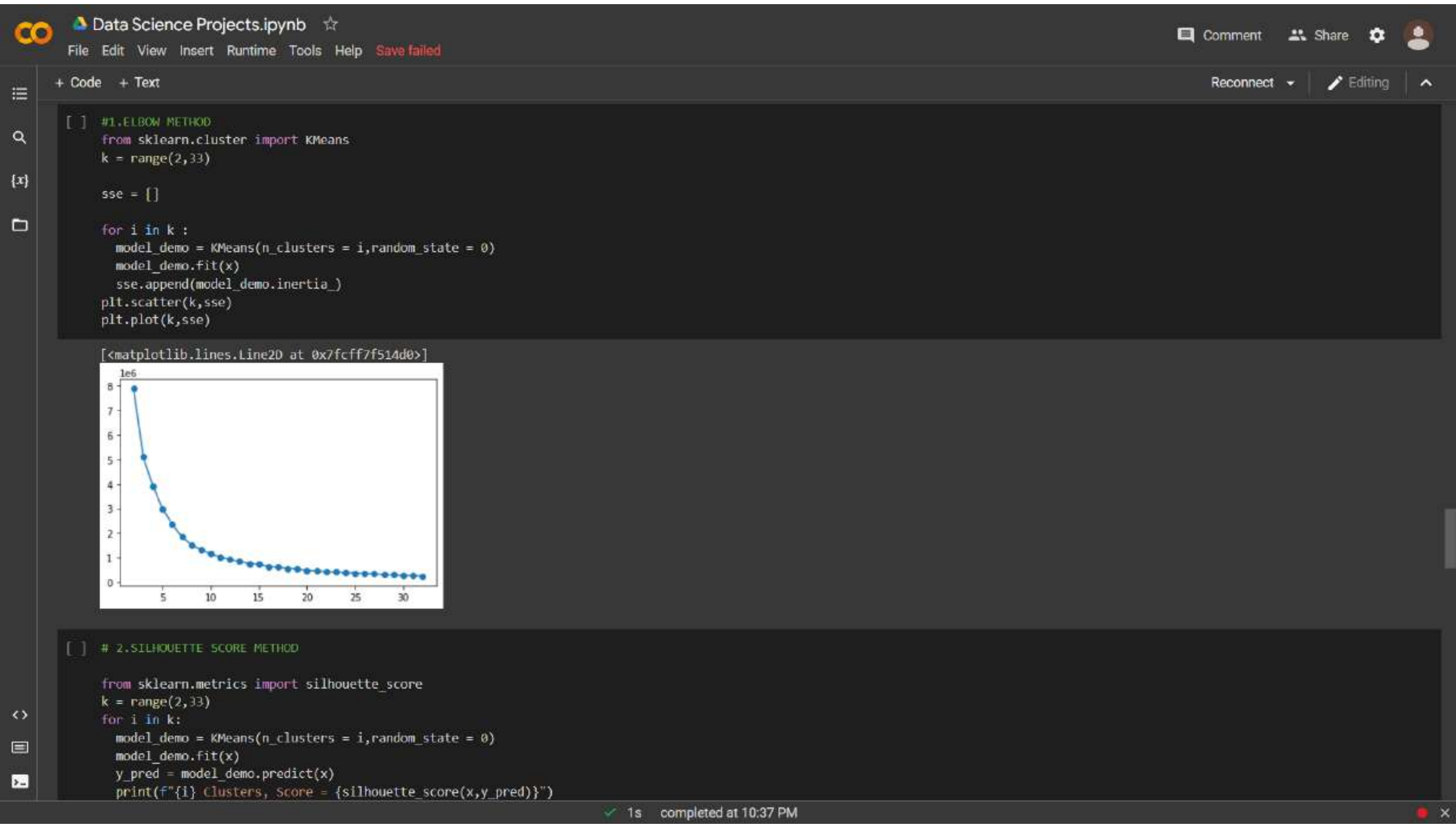
```
<matplotlib.collections.PathCollection at 0x7fcff987b050>
```

```
[ ] #Now Finding out the number of clusters(k)
    # Total number of points = 1030

    np.sqrt(1030)

    # k value ranges from 2 - 32
```

```
32.09361307176243
```

```
[ ] # Now we have to find No. of clusters(k)
```

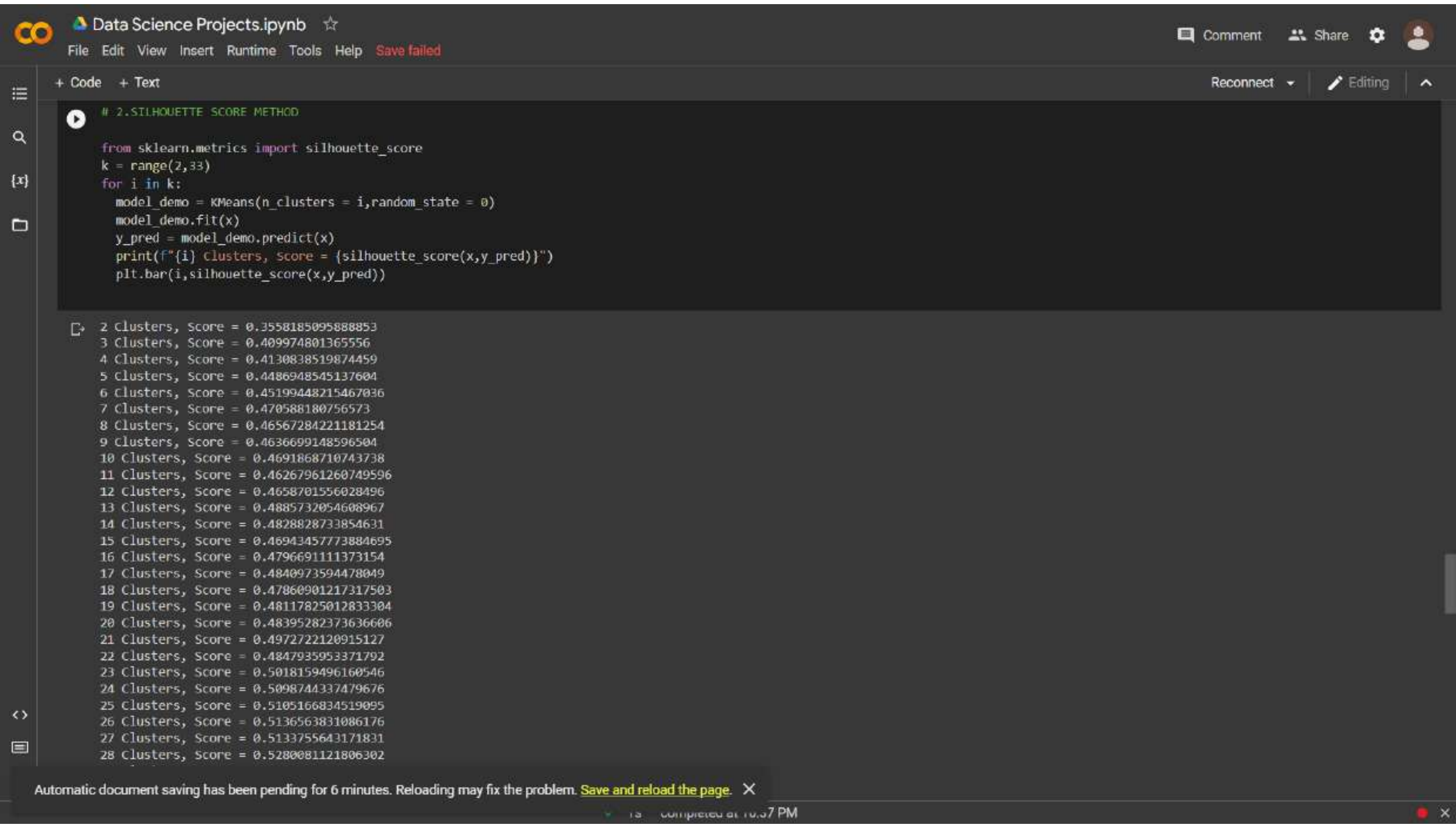
```
[ ] #1.ELBOW METHOD
    from sklearn.cluster import KMeans
    k = range(2,33)

    sse = []
```



```
#1.ELBOW METHOD
from sklearn.cluster import KMeans
k = range(2,33)

sse = []

for i in k :
  model_demo = KMeans(n_clusters = i,random_state = 0)
  model_demo.fit(x)
  sse.append(model_demo.inertia_)
plt.scatter(k,sse)
plt.plot(k,sse)
```

```
[<matplotlib.lines.Line2D at 0x7fcff7f514d0>]
```

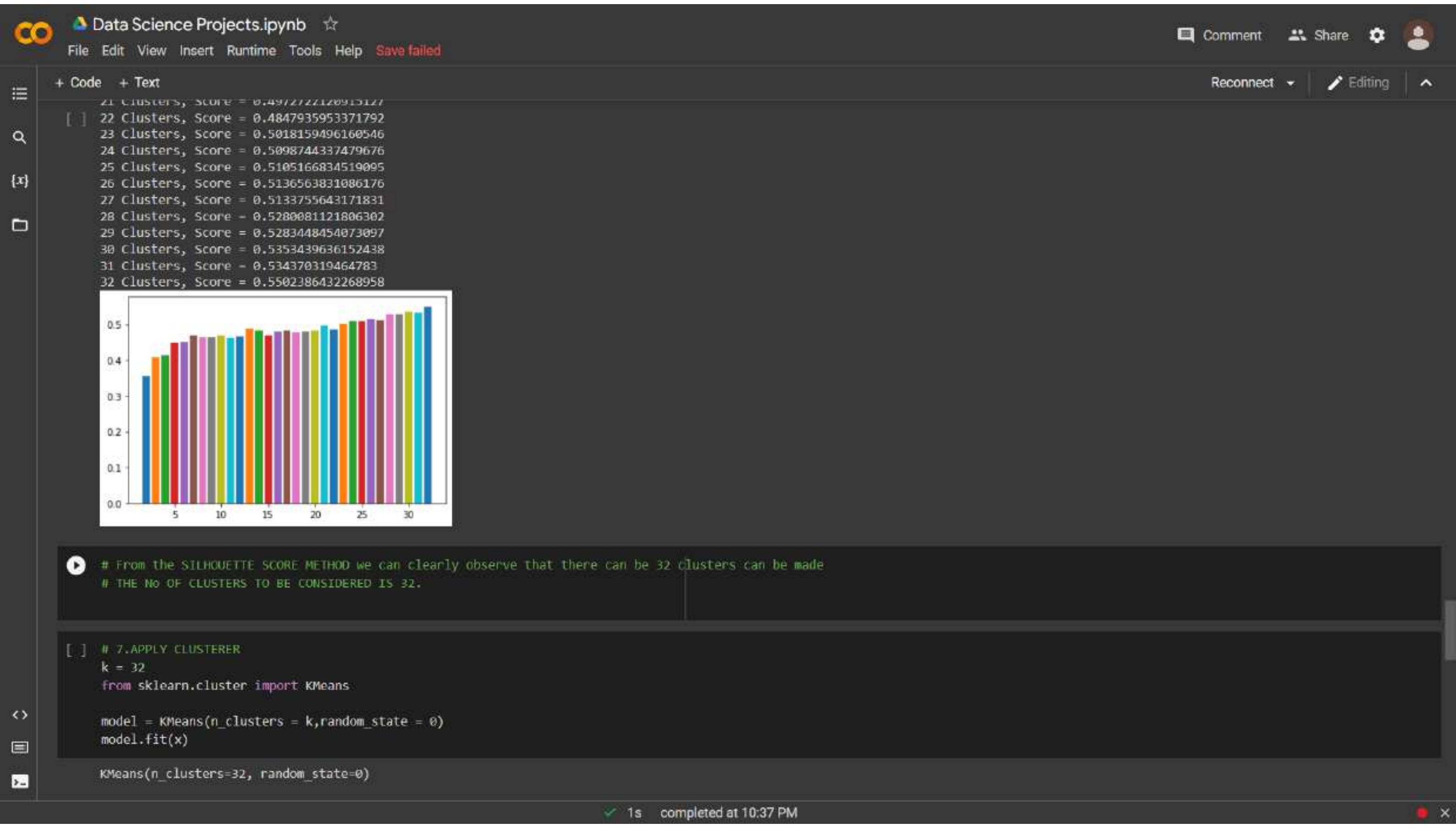
```
# 2.SILHOUETTE SCORE METHOD

from sklearn.metrics import silhouette_score
k = range(2,33)
for i in k:
  model_demo = KMeans(n_clusters = i,random_state = 0)
  model_demo.fit(x)
  y_pred = model_demo.predict(x)
  print(f"{i} Clusters, Score = {silhouette_score(x,y_pred)}")
```


Data Science Projects.ipynb
File Edit View Insert Runtime Tools Help Save failed
Comment
Share
+ Code + Text
Reconnect
Editing

```
# 2.SILHOUETTE SCORE METHOD

from sklearn.metrics import silhouette_score
k = range(2,33)
for i in k:
  model_demo = KMeans(n_clusters = i,random_state = 0)
  model_demo.fit(x)
  y_pred = model_demo.predict(x)
  print(f"{i} Clusters, Score = {silhouette_score(x,y_pred)}")
  plt.bar(i,silhouette_score(x,y_pred))
```

```
2 Clusters, Score = 0.3558185095888853
3 Clusters, Score = 0.409974801365556
4 Clusters, Score = 0.4130838519874459
5 Clusters, Score = 0.4486948545137604
6 Clusters, Score = 0.45199448215467036
7 Clusters, Score = 0.470588180756573
8 Clusters, Score = 0.46567284221181254
9 Clusters, Score = 0.4636699148596504
10 Clusters, Score = 0.4691868710743738
11 Clusters, Score = 0.46267961260749596
12 Clusters, Score = 0.4658701556028496
13 Clusters, Score = 0.4885732054608967
14 Clusters, Score = 0.482882873385463l
15 Clusters, Score = 0.46943457773884695
16 Clusters, Score = 0.4796691111373154
17 Clusters, Score = 0.4840973594478049
18 Clusters, Score = 0.47860901217317503
19 Clusters, Score = 0.48117825012833304
20 Clusters, Score = 0.48395282373636606
21 Clusters, Score = 0.4972722120915127
22 Clusters, Score = 0.4847935953371792
23 Clusters, Score = 0.5018159496160546
24 Clusters, Score = 0.5098744337479676
25 Clusters, Score = 0.5105166834519095
26 Clusters, Score = 0.5136563831086176
27 Clusters, Score = 0.5133755643171831
28 Clusters, Score = 0.5280081121806302
```

Automatic document saving has been pending for 6 minutes. Reloading may fix the problem. Save and reload the page.



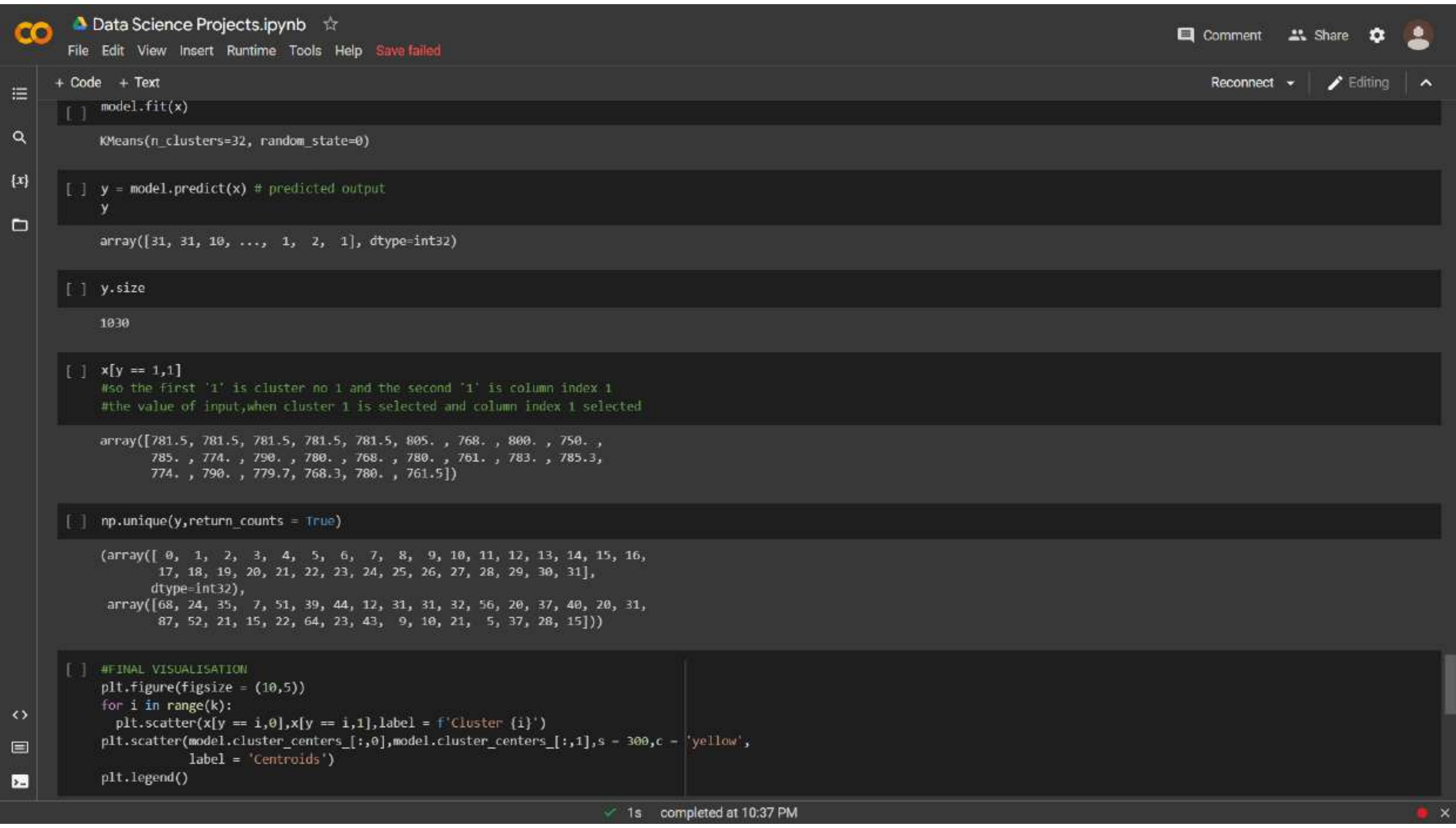
```
21 Clusters, Score = 0.4972722120915127
22 Clusters, Score = 0.4847935953371792
23 Clusters, Score = 0.5018159496160546
24 Clusters, Score = 0.5098744337479676
25 Clusters, Score = 0.5105166834519095
26 Clusters, Score = 0.5136563831086176
27 Clusters, Score = 0.5133755643171831
28 Clusters, Score = 0.5280081121806302
29 Clusters, Score = 0.5283448454073097
30 Clusters, Score = 0.5353439636152438
31 Clusters, Score = 0.534370319464783
32 Clusters, Score = 0.5502386432268958
```

```
# From the SILHOUETTE SCORE METHOD we can clearly observe that there can be 32 clusters can be made
# THE No OF CLUSTERS TO BE CONSIDERED IS 32.
```

```
# 7.APPLY CLUSTERER
k = 32
from sklearn.cluster import KMeans

model = KMeans(n_clusters = k,random_state = 0)
model.fit(x)
```

```
KMeans(n_clusters=32, random_state=0)
```

1s completed at 10:37 PM


Data Science Projects.ipynb
File Edit View Insert Runtime Tools Help Save failed
Comment Share
+ Code + Text
Reconnect Editing
✓ 1s completed at 10:37 PM

```
model.fit(x)
```

```
KMeans(n_clusters=32, random_state=0)
```

```
y = model.predict(x) # predicted output
y
```

```
array([31, 31, 10, ...,  1,  2,  1], dtype=int32)
```

```
y.size
```

```
1030
```

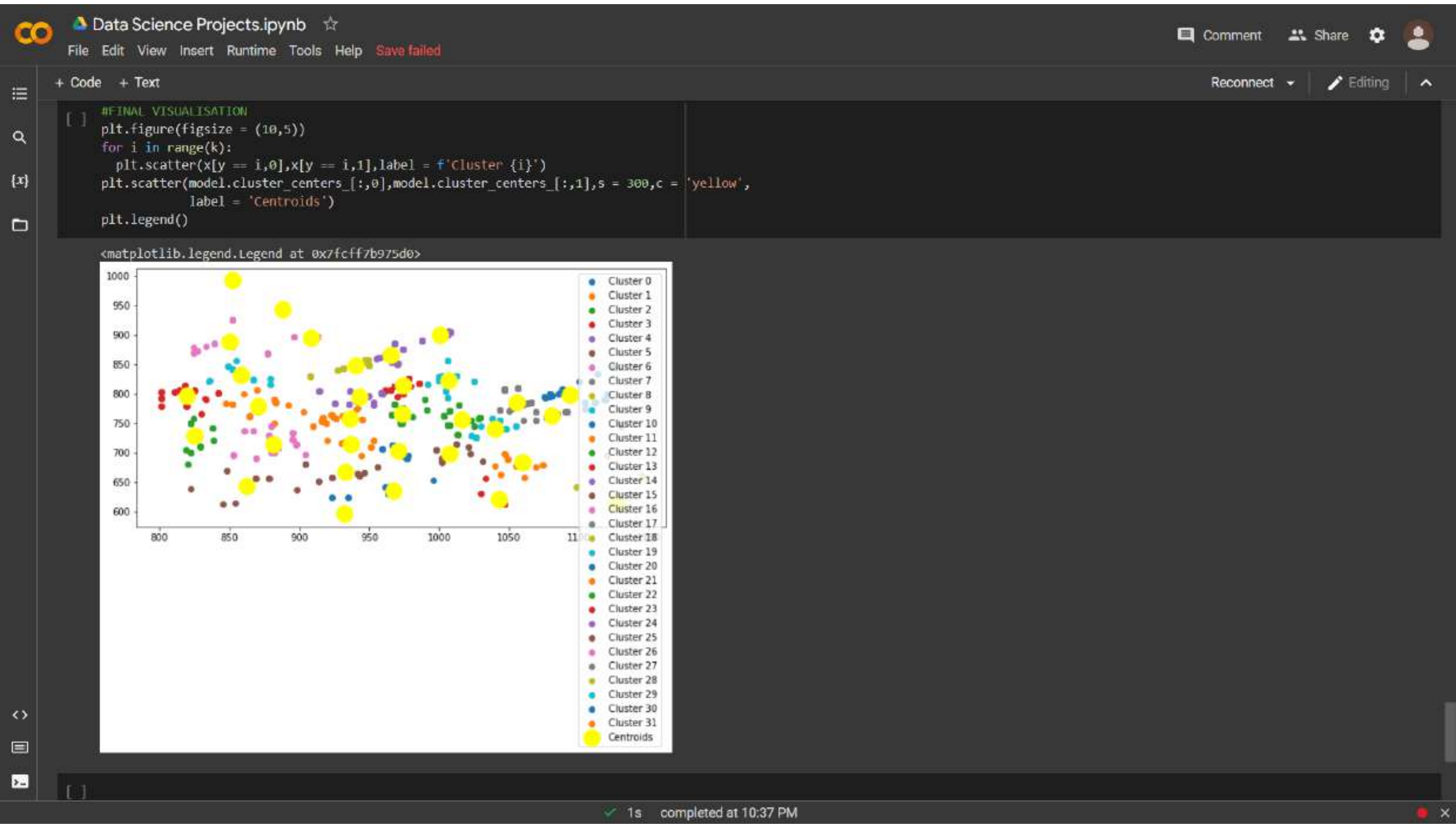
```
x[y == 1,1]
#so the first '1' is cluster no 1 and the second '1' is column index 1
#the value of input,when cluster 1 is selected and column index 1 selected
```

```
array([781.5, 781.5, 781.5, 781.5, 781.5, 805. , 768. , 800. , 750. ,
       785. , 774. , 790. , 780. , 768. , 780. , 761. , 783. , 785.3,
       774. , 790. , 779.7, 768.3, 780. , 761.5])
```

```
np.unique(y,return_counts = True)
```

```
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
        17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31],
       dtype=int32),
 array([68, 24, 35,  7, 51, 39, 44, 12, 31, 31, 32, 56, 20, 37, 40, 20, 31,
        87, 52, 21, 15, 22, 64, 23, 43,  9, 10, 21,  5, 37, 28, 15]))
```

```
#FINAL VISUALISATION
plt.figure(figsize = (10,5))
for i in range(k):
  plt.scatter(x[y == i,0],x[y == i,1],label = f'Cluster {i}')
plt.scatter(model.cluster_centers_[:,0],model.cluster_centers_[:,1],s = 300,c = 'yellow',
            label = 'Centroids')
plt.legend()
```



```
#FINAL VISUALISATION
plt.figure(figsize = (10,5))
for i in range(k):
  plt.scatter(x[y == i,0],x[y == i,1],label = f'Cluster {i}')
plt.scatter(model.cluster_centers_[:,0],model.cluster_centers_[:,1],s = 300,c = 'yellow',
            label = 'Centroids')
plt.legend()
```

```
<matplotlib.legend.Legend at 0x7fcff7b975d0>
```