practice

April 5, 2024

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import linear_model , tree
from sklearn.linear_model import LinearRegression , LogisticRegression
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.cluster import KMeans
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score , precision_score , recall_score ,␣
↪f1_score ,classification_report , confusion_matrix
from sklearn.preprocessing import LabelEncoder , MinMaxScaler
```

[2]:
```
data = sns.load_dataset('iris')
data.head(10)
```

[2]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa
5	5.4	3.9	1.7	0.4	setosa
6	4.6	3.4	1.4	0.3	setosa
7	5.0	3.4	1.5	0.2	setosa
8	4.4	2.9	1.4	0.2	setosa
9	4.9	3.1	1.5	0.1	setosa

[3]:
```
data['species'].unique()
```

[3]:
```
array(['setosa', 'versicolor', 'virginica'], dtype=object)
```

[4]:
```
data['new_species'] = data['species'].map({'setosa':0,'versicolor':
↪1,'virginica':2})
```

[5]:
```
data.head()
```

[5]:

	sepal_length	sepal_width	petal_length	petal_width	species	new_species
0	5.1	3.5	1.4	0.2	setosa	0
1	4.9	3.0	1.4	0.2	setosa	0
2	4.7	3.2	1.3	0.2	setosa	0
3	4.6	3.1	1.5	0.2	setosa	0
4	5.0	3.6	1.4	0.2	setosa	0

[6]:
```
data['new_species'].unique()
```

[6]:
```
array([0, 1, 2], dtype=int64)
```

Linear Regression

[7]:
```
# from sklearn.model_selection import train_test_split
x = data.iloc[:,0:4].values
x_train,x_test,y_train,y_test = train_test_split(x,data.new_species,test_size=0.
↪3)
```

[8]:
```
# from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(x_train,y_train)
```

[8]:
```
LinearRegression()
```

[9]:
```
y_pred = model.predict(x_test)
y_pred
```

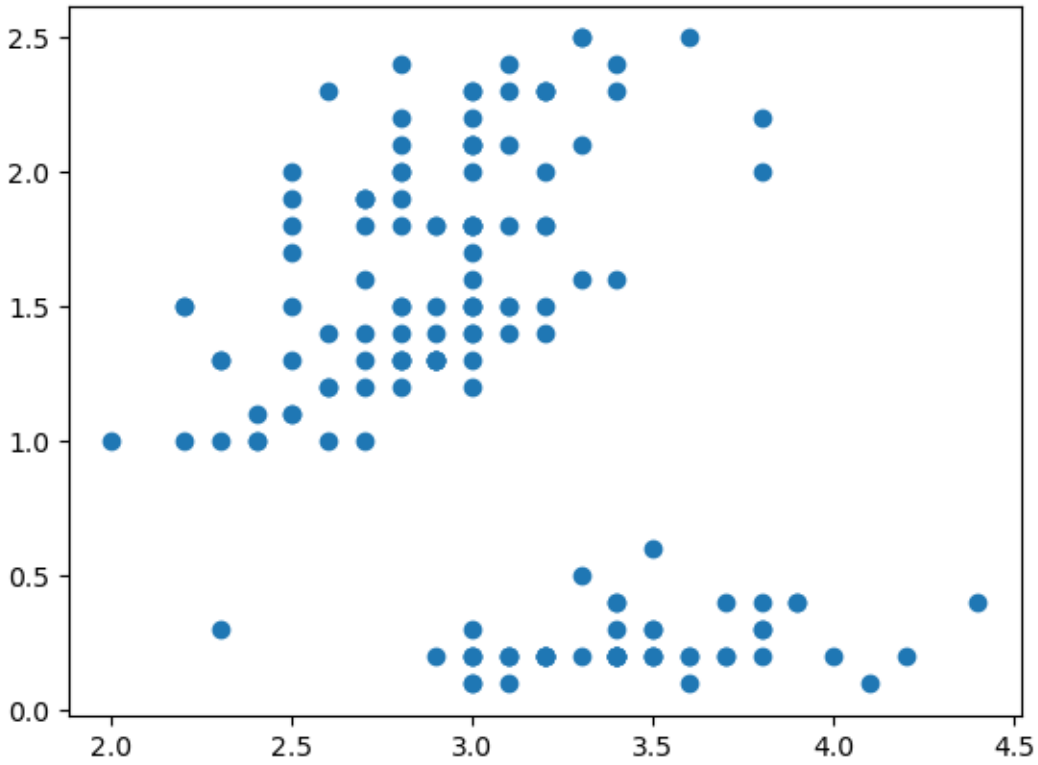
[9]:
```
array([ 1.95801692,  0.03856871,  1.73686464,  1.42439161, -0.18343984,
       -0.02837116, -0.0670467 ,  1.76507489,  1.79697232,  0.02990067,
        1.27769659,  0.96891789,  1.86182515,  1.89661181,  1.18329373,
        1.02748494,  1.24799521, -0.03539331,  0.01564461,  1.55158677,
        1.75760068,  1.17397111,  1.18593469,  1.16524981,  1.59844602,
        0.97723747, -0.1850354 ,  2.07659979, -0.0292157 , -0.06909514,
       -0.13977254, -0.0879836 ,  1.73646209,  0.04111323,  1.80144995,
        1.31184041,  2.20305401,  1.20796816, -0.08309626,  1.36911003,
        1.31821888, -0.15636851,  1.79099723,  0.0378286 ,  1.19445512])
```

[10]:
```
model.score(x_test,y_test)
```

[10]:
```
0.9389893325142514
```

[11]:
```
plt.scatter(data['sepal_width'],data['petal_width'])
```

[11]:
```
<matplotlib.collections.PathCollection at 0x29dbf11a610>
```



Logistic Regression

[12]:
```
# from sklearn.model_selection import train_test_split
x = data.iloc[:,0:4].values
x_train,x_test,y_train,y_test = train_test_split(x,data.species,test_size=0.3)
```

[13]:
```
# from sklearn.linear_model import LinearRegression , LogisticRegression
model = LogisticRegression()
model.fit(x_train,y_train)
```

```
c:\Users\Digvijay\anaconda3\Lib\site-
packages\sklearn\linear_model\_logistic.py:458: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
```

[13]:
```
LogisticRegression()
```

[14]:
```
y_pred = model.predict(x_test)
y_pred
```

[14]:
```
array(['virginica', 'setosa', 'setosa', 'setosa', 'virginica',
       'versicolor', 'versicolor', 'versicolor', 'versicolor', 'setosa',
       'versicolor', 'versicolor', 'virginica', 'virginica', 'setosa',
       'setosa', 'virginica', 'versicolor', 'setosa', 'virginica',
       'versicolor', 'virginica', 'virginica', 'virginica', 'versicolor',
       'setosa', 'virginica', 'virginica', 'versicolor', 'virginica',
       'versicolor', 'versicolor', 'versicolor', 'setosa', 'setosa',
       'virginica', 'versicolor', 'virginica', 'setosa', 'setosa',
       'virginica', 'versicolor', 'virginica', 'virginica', 'setosa'],
      dtype=object)
```

[15]:
```
logi_accu = accuracy_score(y_test,y_pred)
logi_accu
```

[15]:
```
0.9777777777777777
```

[16]:
```
model.score(x_test,y_test)
```

[16]:
```
0.9777777777777777
```

[17]:
```
precision_score(y_test,y_pred,average = 'weighted')
```

[17]:
```
0.9792592592592592
```

[18]:
```
recall_score(y_test,y_pred,average = 'weighted')
```

[18]:
```
0.9777777777777777
```

[19]:
```
f1_score(y_test,y_pred,average = 'weighted')
```

[19]:
```
0.9778434592227695
```

[20]:
```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        13
  versicolor       0.93      1.00      0.97        14
   virginica       1.00      0.94      0.97        18

    accuracy                           0.98        45
   macro avg       0.98      0.98      0.98        45
```

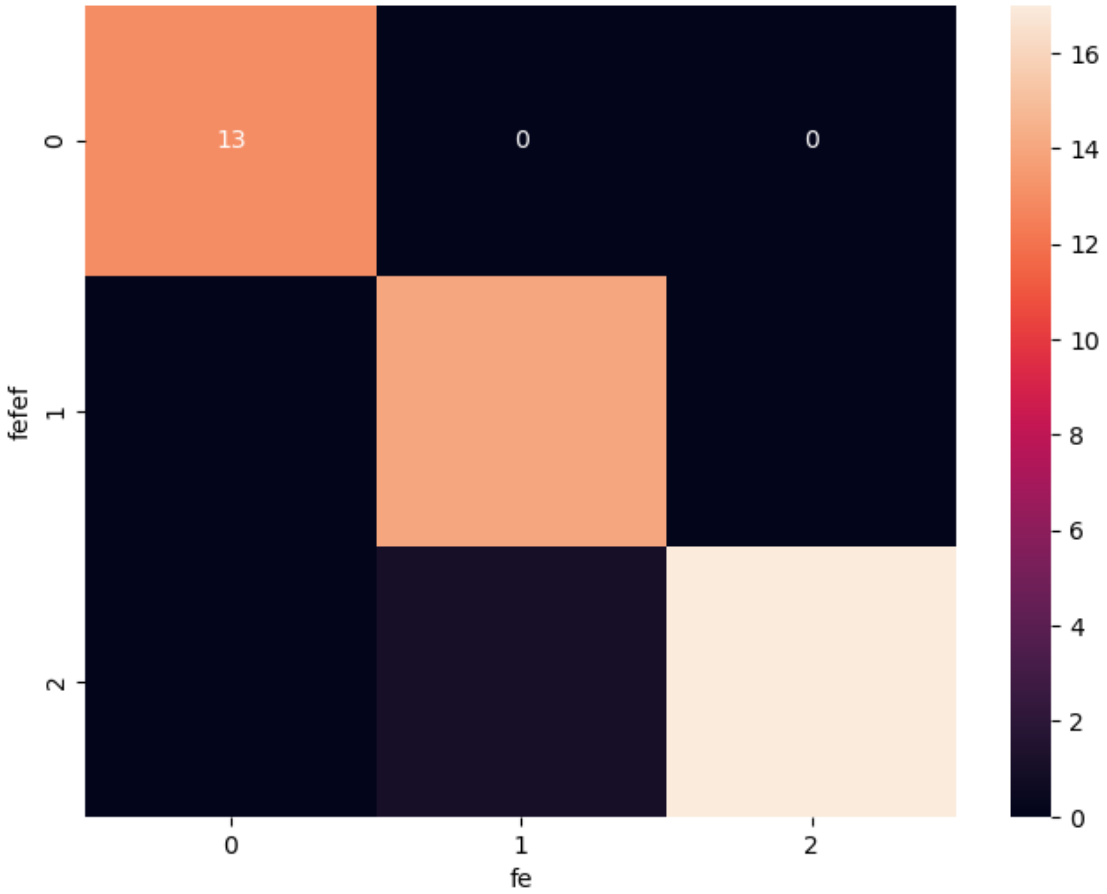
```
weighted avg       0.98      0.98      0.98        45
```

[21]:
```
cm = confusion_matrix(y_test,y_pred)
cm
```

[21]:
```
array([[13,  0,  0],
       [ 0, 14,  0],
       [ 0,  1, 17]], dtype=int64)
```

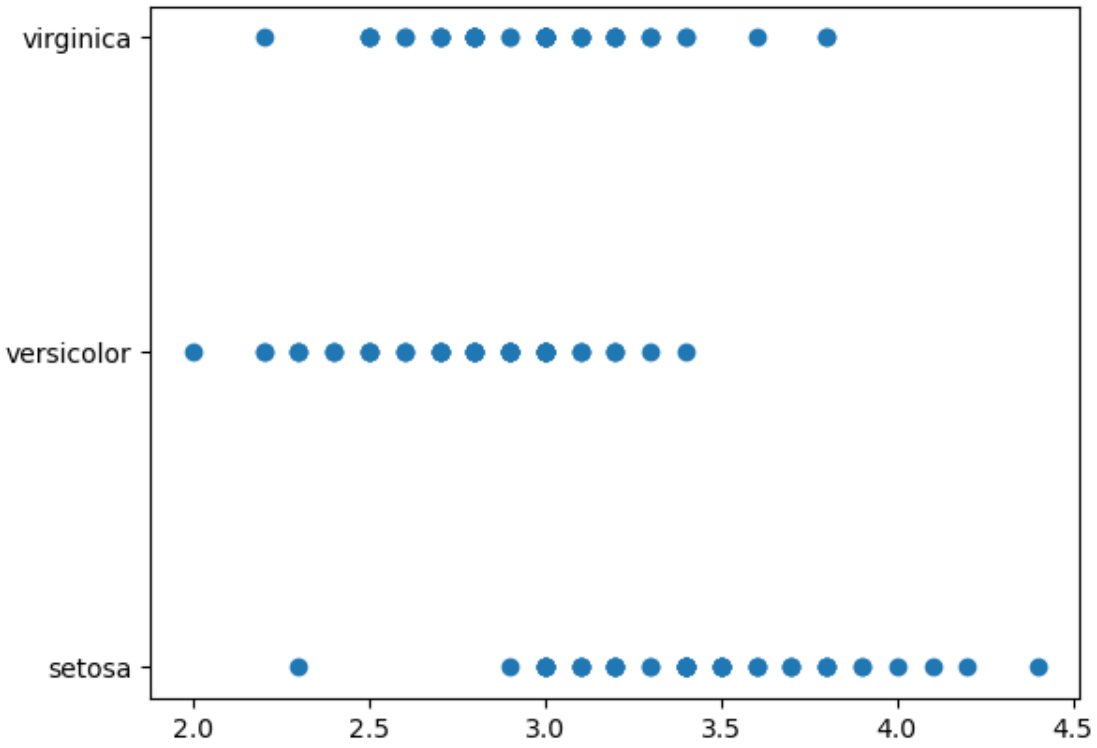
[22]:
```
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True)
plt.xlabel('fe')
plt.ylabel('fefef')
```

[22]:
```
Text(70.72222222222221, 0.5, 'fefef')
```



[23]:
```
plt.scatter(data['sepal_width'],data.species)
```

[23]: <matplotlib.collections.PathCollection at 0x29dbfc8cc50>



SVM

[24]:
```
model = SVC()
model.fit(x_train,y_train)
```

[24]: SVC()

[25]:
```
y_pred = model.predict(x_test)
y_pred
```

[25]:
```
array(['virginica', 'setosa', 'setosa', 'setosa', 'virginica',
       'versicolor', 'versicolor', 'versicolor', 'versicolor', 'setosa',
       'versicolor', 'versicolor', 'virginica', 'virginica', 'setosa',
       'setosa', 'virginica', 'versicolor', 'setosa', 'virginica',
       'versicolor', 'virginica', 'virginica', 'virginica', 'versicolor',
       'setosa', 'virginica', 'virginica', 'versicolor', 'virginica',
       'versicolor', 'versicolor', 'versicolor', 'setosa', 'setosa',
       'virginica', 'versicolor', 'virginica', 'setosa', 'setosa',
       'virginica', 'versicolor', 'versicolor', 'virginica', 'setosa'],
      dtype=object)
```

[26]:
```
svm_accu = accuracy_score(y_test,y_pred)
svm_accu
```

[26]: 0.9555555555555556

[27]:
```
model.score(x_test,y_test)
```

[27]: 0.9555555555555556

[28]:
```
precision_score(y_test,y_pred,average='weighted')
```

[28]: 0.9611111111111111

[29]:
```
recall_score(y_test,y_pred,average='weighted')
```

[29]: 0.9555555555555556

[30]:
```
f1_score(y_test,y_pred,average='weighted')
```

[30]: 0.9557298474945534

[31]:
```
print(classification_report(y_test,y_pred))
```

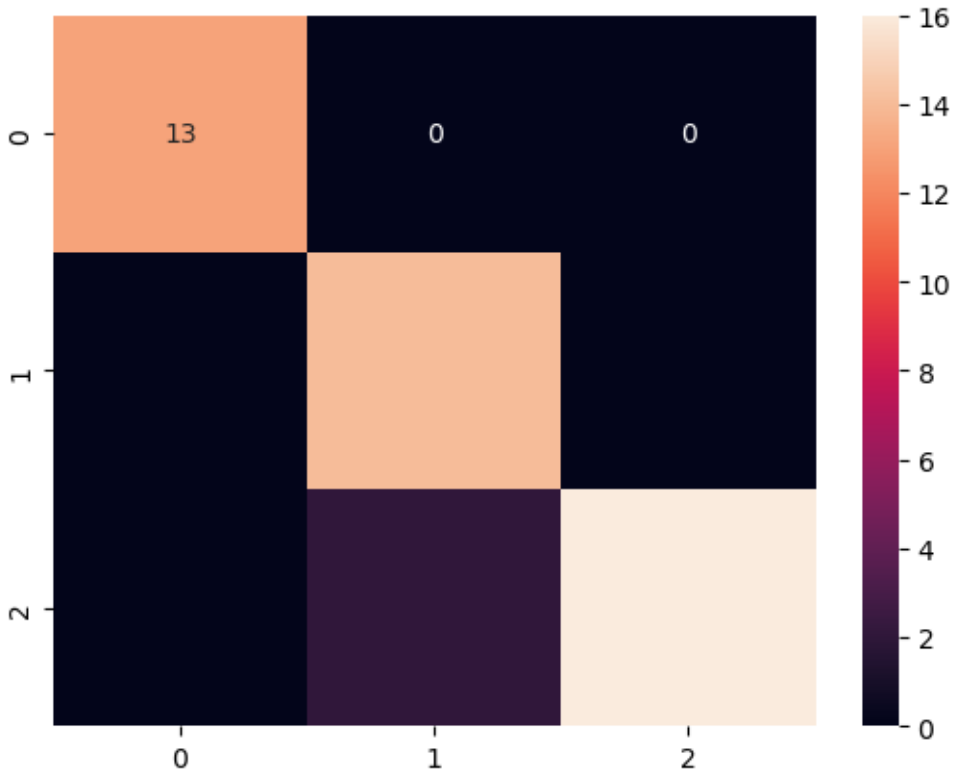
```
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        13
  versicolor       0.88      1.00      0.93        14
   virginica       1.00      0.89      0.94        18

    accuracy                           0.96        45
   macro avg       0.96      0.96      0.96        45
weighted avg       0.96      0.96      0.96        45
```

[32]:
```
cm = confusion_matrix(y_test,y_pred)
cm
```

[32]:
```
array([[13,  0,  0],
       [ 0, 14,  0],
       [ 0,  2, 16]], dtype=int64)
```

[33]:
```
sns.heatmap(cm,annot=True)
```

[33]: <Axes: >



Decision Tree

[34]:
```
model = DecisionTreeClassifier()
model.fit(x_train,y_train)
```

[34]:
```
DecisionTreeClassifier()
```

[35]:
```
y_pred = model.predict(x_test)
y_pred
```

[35]:
```
array(['virginica', 'setosa', 'setosa', 'setosa', 'virginica',
       'versicolor', 'versicolor', 'versicolor', 'versicolor', 'setosa',
       'versicolor', 'versicolor', 'virginica', 'virginica', 'setosa',
       'setosa', 'virginica', 'versicolor', 'setosa', 'virginica',
       'versicolor', 'virginica', 'virginica', 'virginica', 'versicolor',
       'setosa', 'virginica', 'virginica', 'versicolor', 'virginica',
       'versicolor', 'virginica', 'versicolor', 'setosa', 'setosa',
       'virginica', 'versicolor', 'virginica', 'setosa', 'setosa',
       'virginica', 'versicolor', 'virginica', 'virginica', 'setosa'],
      dtype=object)
```

[36]:
```
dec_accu = accuracy_score(y_test,y_pred)
dec_accu
```

[36]: 1.0

[37]:
```
model.score(x_test,y_test)
```

[37]: 1.0

[38]:
```
precision_score(y_test,y_pred,average='weighted')
```

[38]: 1.0

[39]:
```
recall_score(y_test,y_pred,average='weighted')
```

[39]: 1.0

[40]:
```
f1_score(y_test,y_pred,average='weighted')
```

[40]: 1.0

[41]:
```
print(classification_report(y_test,y_pred))
```

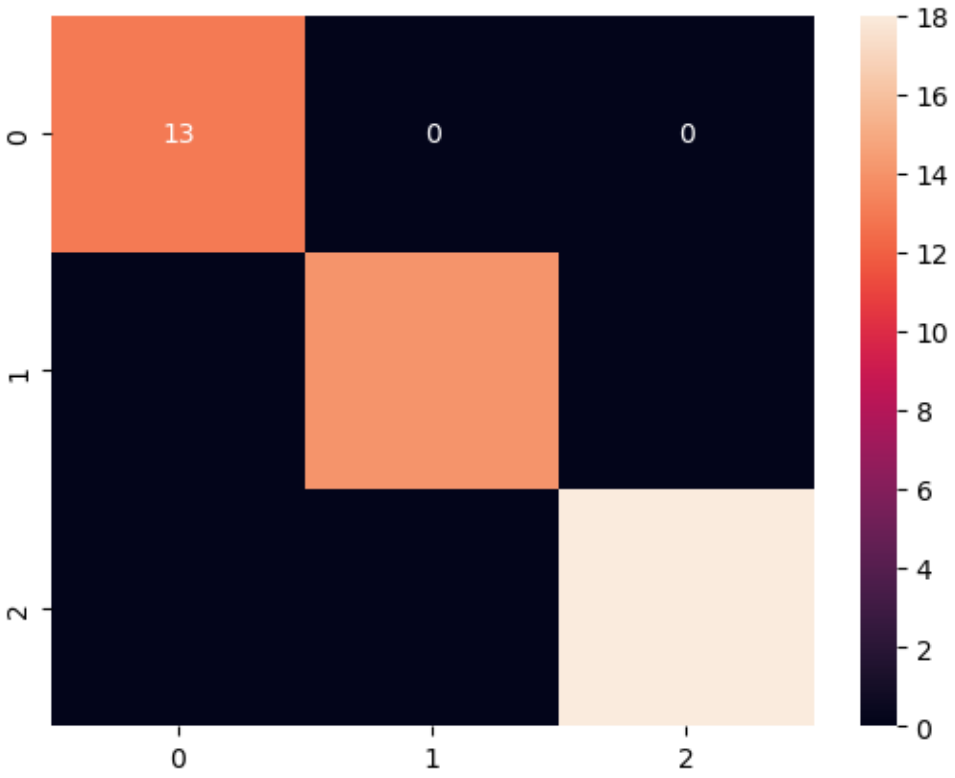
```
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        13
  versicolor       1.00      1.00      1.00        14
   virginica       1.00      1.00      1.00        18

    accuracy                           1.00        45
   macro avg       1.00      1.00      1.00        45
weighted avg       1.00      1.00      1.00        45
```

[42]:
```
cm = confusion_matrix(y_test,y_pred)
cm
```

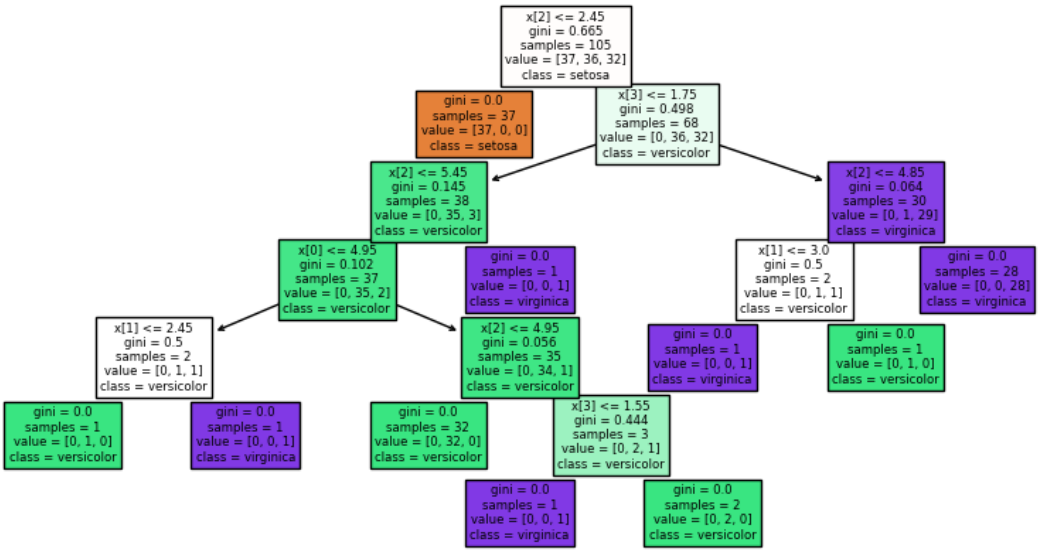
[42]:
```
array([[13,  0,  0],
       [ 0, 14,  0],
       [ 0,  0, 18]], dtype=int64)
```

[43]:
```
sns.heatmap(cm,annot=True)
```

[43]: <Axes: >



[44]:
```
var = ['setosa','versicolor','virginica']
plt.figure(figsize=(10,5))
tree.plot_tree(model,class_names=var,filled=True)
plt.show()
```



KNN

[45]:
```
model = KNeighborsClassifier(n_neighbors=2)
model.fit(x_train,y_train)
```

[45]:
```
KNeighborsClassifier(n_neighbors=2)
```

[46]:
```
y_pred = model.predict(x_test)
y_pred
```

[46]:
```
array(['versicolor', 'setosa', 'setosa', 'setosa', 'virginica',
       'versicolor', 'versicolor', 'versicolor', 'versicolor', 'setosa',
       'versicolor', 'versicolor', 'virginica', 'virginica', 'setosa',
       'setosa', 'virginica', 'versicolor', 'setosa', 'virginica',
       'versicolor', 'virginica', 'virginica', 'virginica', 'versicolor',
       'setosa', 'virginica', 'virginica', 'versicolor', 'virginica',
       'versicolor', 'versicolor', 'versicolor', 'setosa', 'setosa',
       'virginica', 'versicolor', 'versicolor', 'setosa', 'setosa',
       'virginica', 'versicolor', 'versicolor', 'virginica', 'setosa'],
      dtype=object)
```

[47]:
```
knn_accu = accuracy_score(y_test,y_pred)
knn_accu
```

[47]:
```
0.9111111111111111
```

[48]:
```
model.score(x_test,y_test)
```

[48]:
```
0.9111111111111111
```

[49]:
```
precision_score(y_test,y_pred,average='weighted')
```

[49]:
```
0.9308641975308641
```

[50]:
```
recall_score(y_test,y_pred,average='weighted')
```

[50]:
```
0.9111111111111111
```

[51]:
```
f1_score(y_test,y_pred,average='weighted')
```

[51]:
```
0.9111111111111111
```

[52]:
```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        13
```

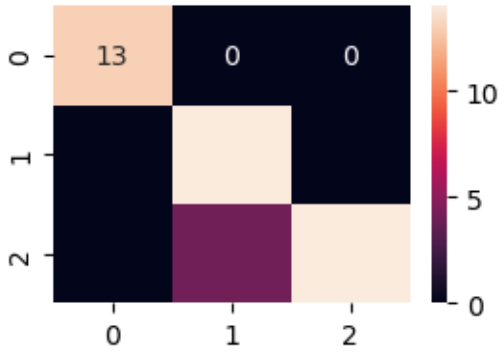
```
  versicolor       0.78      1.00      0.88        14
   virginica       1.00      0.78      0.88        18

    accuracy                           0.91        45
   macro avg       0.93      0.93      0.92        45
weighted avg       0.93      0.91      0.91        45
```

[53]:
```
cm = confusion_matrix(y_test,y_pred)
cm
```

[53]:
```
array([[13,  0,  0],
       [ 0, 14,  0],
       [ 0,  4, 14]], dtype=int64)
```

[54]:
```
plt.figure(figsize=(3,2))
sns.heatmap(cm,annot=True)
```

[54]:
```
<Axes: >
```



K Means (Clustering)

[55]:
```
data = pd.read_csv('income.csv')
data
```

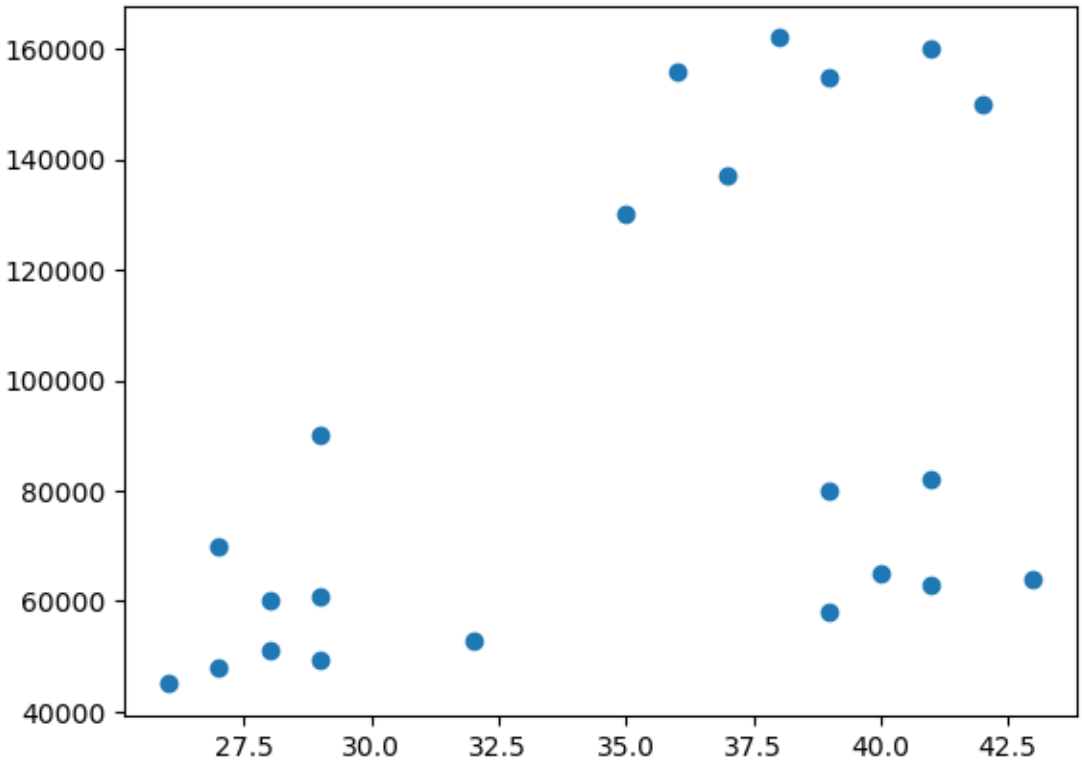
[55]:

	Name	Age	Income($)
0	Rob	27	70000
1	Michael	29	90000
2	Mohan	29	61000
3	Ismail	28	60000
4	Kory	42	150000
5	Gautam	39	155000
6	David	41	160000
7	Andrea	38	162000

```
8       Brad   36     156000
9   Angelina   35     130000
10    Donald   37     137000
11       Tom   26      45000
12    Arnold   27      48000
13     Jared   28      51000
14     Stark   29      49500
15    Ranbir   32      53000
16    Dipika   40      65000
17  Priyanka   41      63000
18      Nick   43      64000
19      Alia   39      80000
20       Sid   41      82000
21     Abdul   39      58000
```

[56]:
```
plt.scatter(data.Age,data[['Income($)']])
```

[56]:
```
<matplotlib.collections.PathCollection at 0x29dc0174c50>
```



[57]:
```
model = KMeans(n_clusters=3)
cluster = model.fit_predict(data[['Age','Income($)']])
cluster
```

```
c:\Users\Digvijay\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:870:
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
c:\Users\Digvijay\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1382:
UserWarning: KMeans is known to have a memory leak on Windows with MKL, when
there are less chunks than available threads. You can avoid it by setting the
environment variable OMP_NUM_THREADS=1.
  warnings.warn(
```

[57]:
```
array([0, 0, 2, 2, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 0, 0, 2])
```

[58]:
```
data['cluster'] = cluster
data.head()
```

[58]:

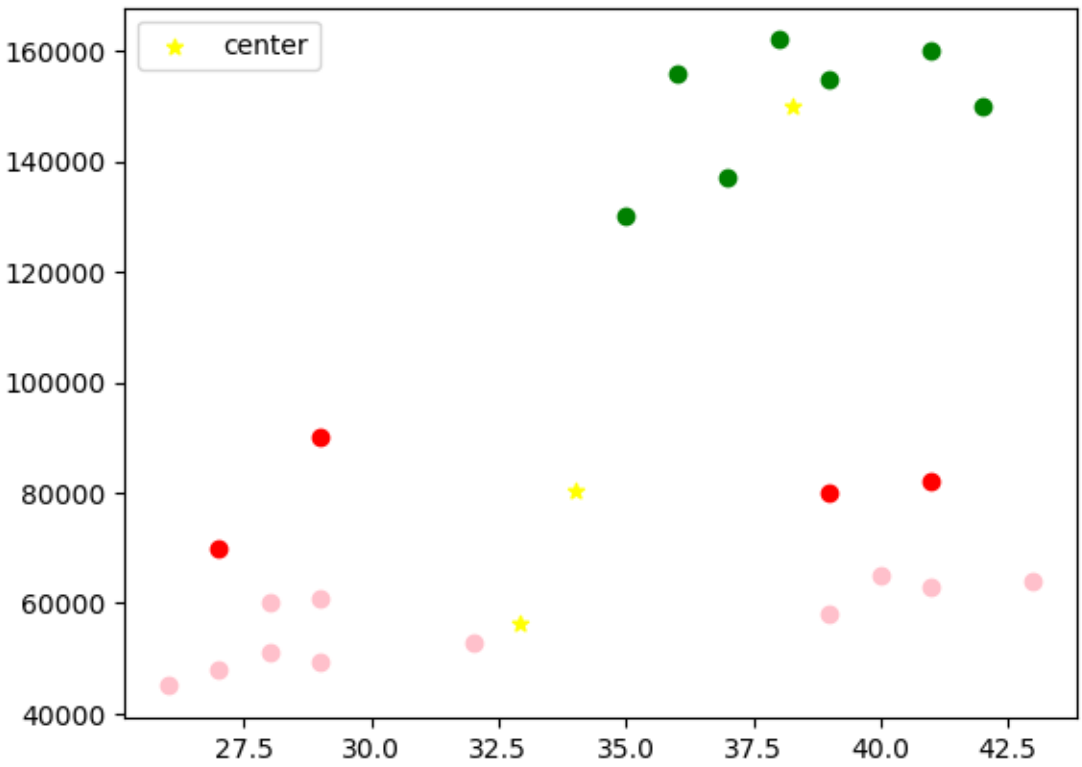
	Name	Age	Income($)	cluster
0	Rob	27	70000	0
1	Michael	29	90000	0
2	Mohan	29	61000	2
3	Ismail	28	60000	2
4	Kory	42	150000	1

[59]:
```
model.cluster_centers_
```

[59]:
```
array([[3.40000000e+01, 8.05000000e+04],
       [3.82857143e+01, 1.50000000e+05],
       [3.29090909e+01, 5.61363636e+04]])
```

[60]:
```
df0 = data[data.cluster == 0]
df1 = data[data.cluster == 1]
df2 = data[data.cluster == 2]
# print(df0)
# print(df1)
# print(df2)
plt.scatter(df0.Age,df0['Income($)'],color='red')
plt.scatter(df1.Age,df1['Income($)'],color='green')
plt.scatter(df2.Age,df2['Income($)'],color='pink')
plt.scatter(model.cluster_centers_[:,0],model.cluster_centers_[:,1],color =␣
 ↪'yellow' , marker='*',label='center')
plt.legend()
```

[60]:
```
<matplotlib.legend.Legend at 0x29dc1304a10>
```



[61]:
```
scaler = MinMaxScaler()
scaler.fit(data[['Income($)']])
```

[61]:
```
MinMaxScaler()
```

[62]:
```
data['Income($)'] = scaler.transform(data[['Income($)']])
data.head()
```

[62]:

	Name	Age	Income($)	cluster
0	Rob	27	0.213675	0
1	Michael	29	0.384615	0
2	Mohan	29	0.136752	2
3	Ismail	28	0.128205	2
4	Kory	42	0.897436	1

[63]:
```
scaler.fit(data[['Age']])
data['Age'] = scaler.transform(data[['Age']])
data.head()
```

[63]:

	Name	Age	Income($)	cluster
0	Rob	0.058824	0.213675	0
1	Michael	0.176471	0.384615	0

```
2    Mohan  0.176471   0.136752        2
3   Ismail  0.117647   0.128205        2
4     Kory  0.941176   0.897436        1
```

[64]:
```
model = KMeans(n_clusters=3)
cluster = model.fit_predict(data[['Age','Income($)']])
cluster
```

```
c:\Users\Digvijay\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:870:
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
c:\Users\Digvijay\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1382:
UserWarning: KMeans is known to have a memory leak on Windows with MKL, when
there are less chunks than available threads. You can avoid it by setting the
environment variable OMP_NUM_THREADS=1.
  warnings.warn(
```

[64]:
```
array([0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 2, 2, 2, 2, 2, 2])
```

[65]:
```
data['cluster'] = cluster
data.head()
```

[65]:
```
      Name       Age  Income($)  cluster
0      Rob  0.058824   0.213675        0
1  Michael  0.176471   0.384615        0
2    Mohan  0.176471   0.136752        0
3   Ismail  0.117647   0.128205        0
4     Kory  0.941176   0.897436        1
```

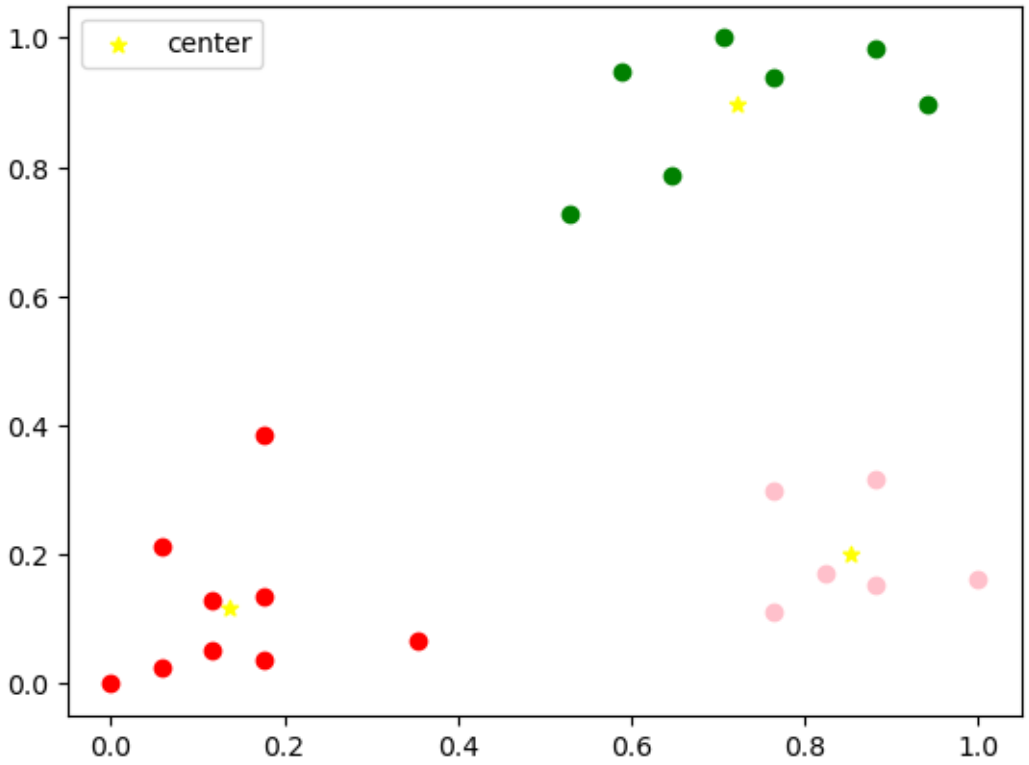
[66]:
```
model.cluster_centers_
```

[66]:
```
array([[0.1372549 , 0.11633428],
       [0.72268908, 0.8974359 ],
       [0.85294118, 0.2022792 ]])
```

[67]:
```
df0 = data[data.cluster == 0]
df1 = data[data.cluster == 1]
df2 = data[data.cluster == 2]
# print(df0)
# print(df1)
# print(df2)
plt.scatter(df0.Age,df0['Income($)'],color='red')
plt.scatter(df1.Age,df1['Income($)'],color='green')
plt.scatter(df2.Age,df2['Income($)'],color='pink')
plt.scatter(model.cluster_centers_[:,0],model.cluster_centers_[:,1],color =␣
 ↪'yellow' , marker='*',label='center')
```

```
plt.legend()
```

[67]: <matplotlib.legend.Legend at 0x29dc1385b50>



[68]:
```
sse = []
k_range = range(1,10)

for k in k_range:
    model = KMeans(n_clusters=k)
    model.fit(data[['Age','Income($)']])
    sse.append(model.inertia_)
```

```
c:\Users\Digvijay\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:870:
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
c:\Users\Digvijay\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1382:
UserWarning: KMeans is known to have a memory leak on Windows with MKL, when
there are less chunks than available threads. You can avoid it by setting the
environment variable OMP_NUM_THREADS=1.
  warnings.warn(
c:\Users\Digvijay\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:870:
```

```
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
c:\Users\Digvijay\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1382:
UserWarning: KMeans is known to have a memory leak on Windows with MKL, when
there are less chunks than available threads. You can avoid it by setting the
environment variable OMP_NUM_THREADS=1.
  warnings.warn(
c:\Users\Digvijay\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:870:
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
c:\Users\Digvijay\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1382:
UserWarning: KMeans is known to have a memory leak on Windows with MKL, when
there are less chunks than available threads. You can avoid it by setting the
environment variable OMP_NUM_THREADS=1.
  warnings.warn(
c:\Users\Digvijay\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:870:
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
c:\Users\Digvijay\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1382:
UserWarning: KMeans is known to have a memory leak on Windows with MKL, when
there are less chunks than available threads. You can avoid it by setting the
environment variable OMP_NUM_THREADS=1.
  warnings.warn(
c:\Users\Digvijay\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:870:
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
c:\Users\Digvijay\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1382:
UserWarning: KMeans is known to have a memory leak on Windows with MKL, when
there are less chunks than available threads. You can avoid it by setting the
environment variable OMP_NUM_THREADS=1.
  warnings.warn(
c:\Users\Digvijay\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:870:
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
c:\Users\Digvijay\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1382:
UserWarning: KMeans is known to have a memory leak on Windows with MKL, when
there are less chunks than available threads. You can avoid it by setting the
environment variable OMP_NUM_THREADS=1.
  warnings.warn(
c:\Users\Digvijay\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:870:
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
```

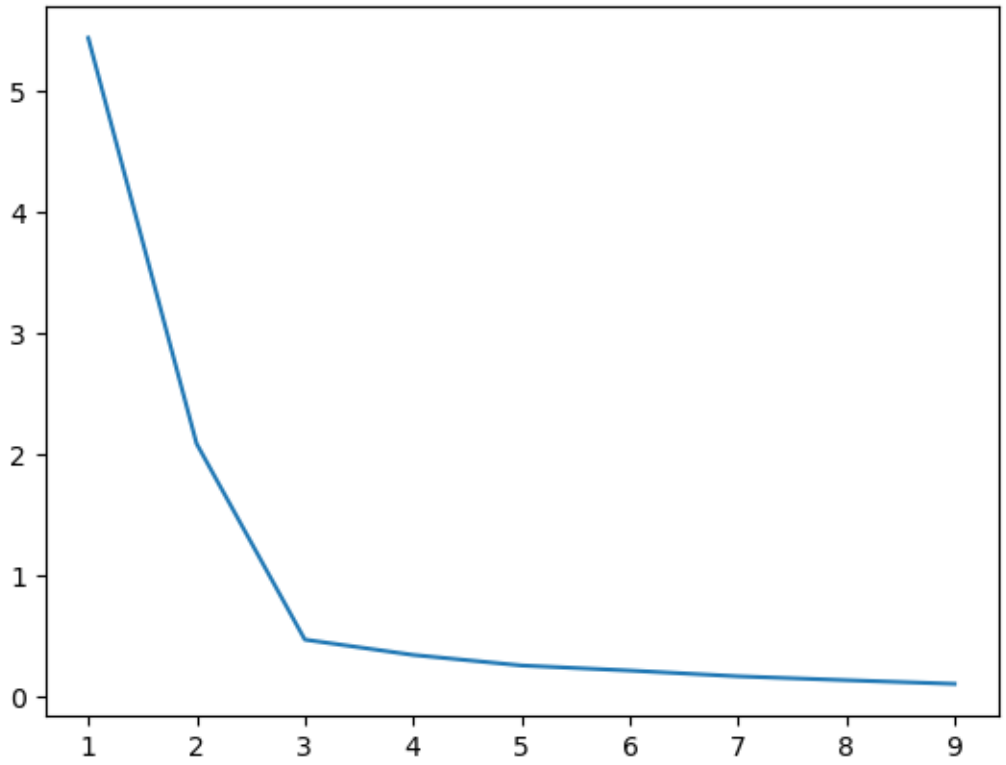
```
c:\Users\Digvijay\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1382:
UserWarning: KMeans is known to have a memory leak on Windows with MKL, when
there are less chunks than available threads. You can avoid it by setting the
environment variable OMP_NUM_THREADS=1.
  warnings.warn(
c:\Users\Digvijay\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:870:
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
c:\Users\Digvijay\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1382:
UserWarning: KMeans is known to have a memory leak on Windows with MKL, when
there are less chunks than available threads. You can avoid it by setting the
environment variable OMP_NUM_THREADS=1.
  warnings.warn(
c:\Users\Digvijay\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:870:
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
c:\Users\Digvijay\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1382:
UserWarning: KMeans is known to have a memory leak on Windows with MKL, when
there are less chunks than available threads. You can avoid it by setting the
environment variable OMP_NUM_THREADS=1.
  warnings.warn(
```

[69]:
```
sse
```

[69]:
```
[5.434011511988179,
 2.091136388699078,
 0.4750783498553096,
 0.34910470944195654,
 0.2621792762345213,
 0.2203764169077067,
 0.17299621932455464,
 0.14090581089405507,
 0.11073569527418642]
```

[70]:
```
plt.plot(k_range,sse)
```

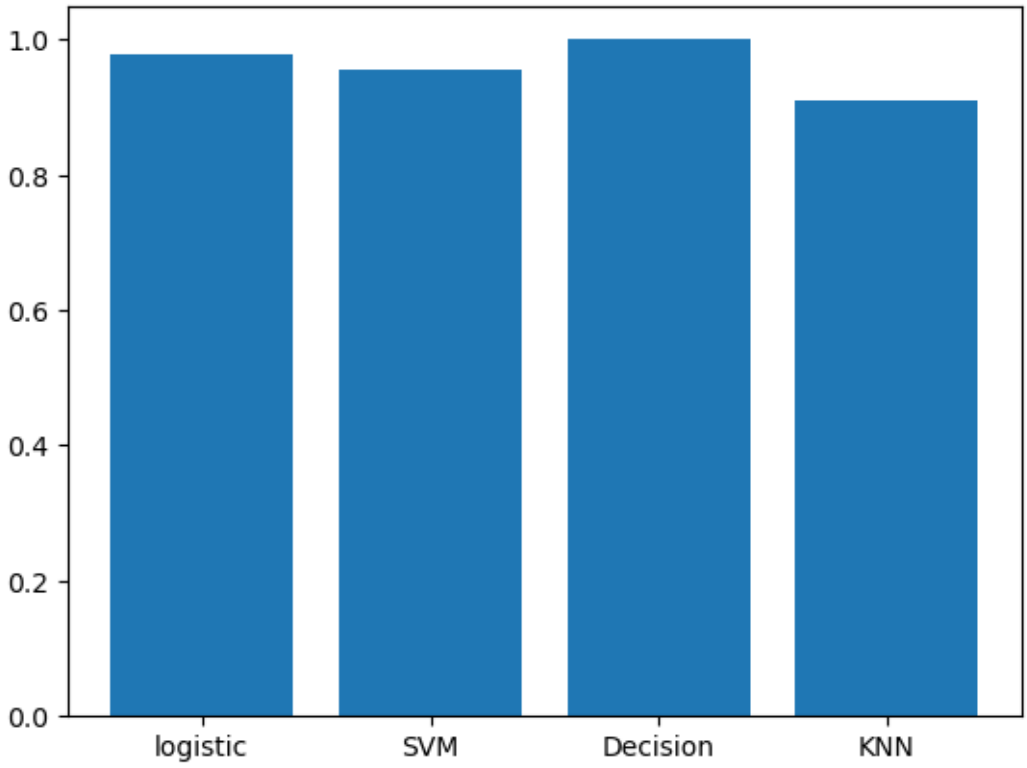
[70]:
```
[<matplotlib.lines.Line2D at 0x29dbf133f50>]
```



Accuracy Bar

[71]:
```
plt.
↪bar(['logistic','SVM','Decision','KNN'],[logi_accu,svm_accu,dec_accu,knn_accu])
```

[71]: <BarContainer object of 4 artists>


1.0
0.8
0.6
0.4
0.2
0.0
logistic
SVM
Decision
KNN

[ ]: