Practical – 6

Shivam Tawari (A-58)

me Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk Editing

Skill Development - Practical-6

Aim:- Solving classification problems using Scikitlearn

Name: Shivam Tawari Roll no: A-58

```
[5] print("Shivam Tawari A-58")
```

```
Shivam Tawari A-58
```

```
[6] import numpy as np
    import pandas as pd
    import matplotlib.pyplot as plt
    import os
    from sklearn.preprocessing import LabelEncoder
    import seaborn as sns
    from sklearn.metrics import accuracy_score
    from sklearn.model_selection import train_test_split
    from sklearn.ensemble import RandomForestClassifier
    from sklearn.linear_model import LogisticRegression
    from sklearn.tree import DecisionTreeClassifier
    from sklearn.svm import SVC
    from sklearn.neighbors import KNeighborsClassifier
```

```
input_path='/content/iris.csv'
df=pd.read_csv(input_path)
df.head()
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

```
[8] df.drop('Id',axis=1,inplace=True)
    df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   SepalLengthCm  150 non-null    float64
 1   SepalWidthCm   150 non-null    float64
 2   PetalLengthCm  150 non-null    float64
 3   PetalWidthCm   150 non-null    float64
```

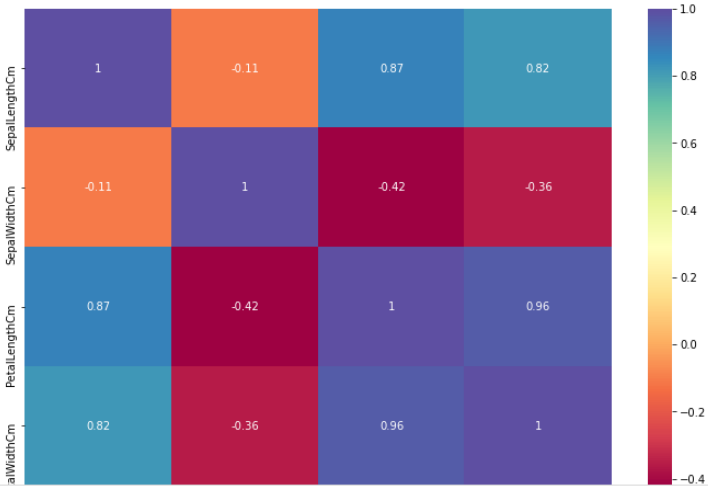
0s completed at 11:04 AM

+ Code + Text

RAM Disk Editing

```
plt.figure(figsize=(12,8))
sns.heatmap(df.corr(),annot=True,cmap=plt.cm.Spectral)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f3685708190>
```

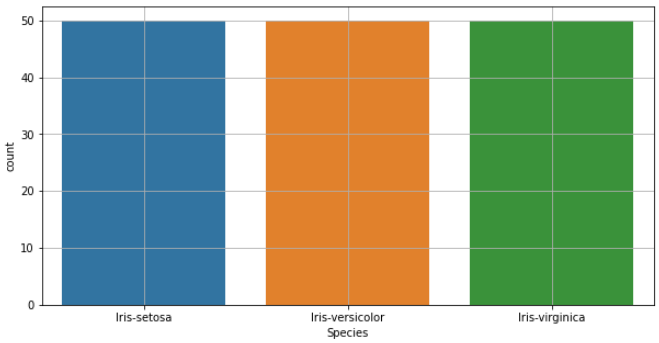


0s completed at 11:04 AM

+ Code + Text

RAM Disk Editing

```
[10] plt.figure(figsize=(10,5))
     sns.countplot(x='Species',data=df)
     plt.grid()
```




```
le=LabelEncoder()
df['Species_type']=le.fit_transform(df['Species'])
```

```
[12] def scatterplot(X,Y):
```

0s completed at 11:04 AM

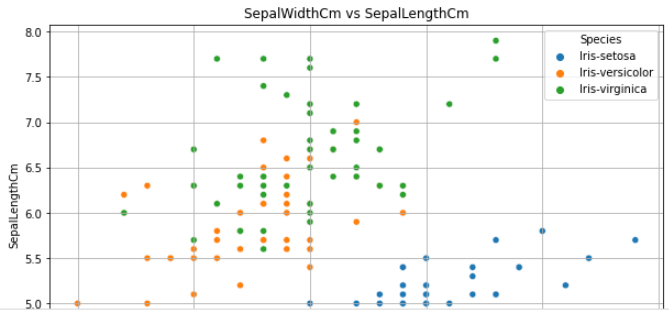
+ Code + Text

RAM Disk

Editing

```
[12] def scatterplot(X,Y):
         sns.scatterplot(data=df,x=X,y=Y,hue='Species')
         plt.title(X+" vs "+Y)
         plt.grid()
         plt.show()
```

```
plt.figure(figsize=(10,6))
scatterplot('SepalWidthCm','SepalLengthCm')
plt.figure(figsize=(10,6))
scatterplot('PetalWidthCm','PetalLengthCm')
plt.tight_layout()
```

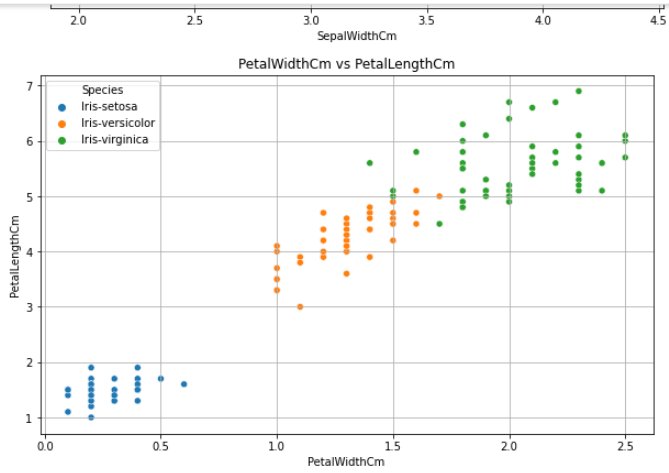


0s completed at 11:04 AM

+ Code + Text

RAM Disk

Editing

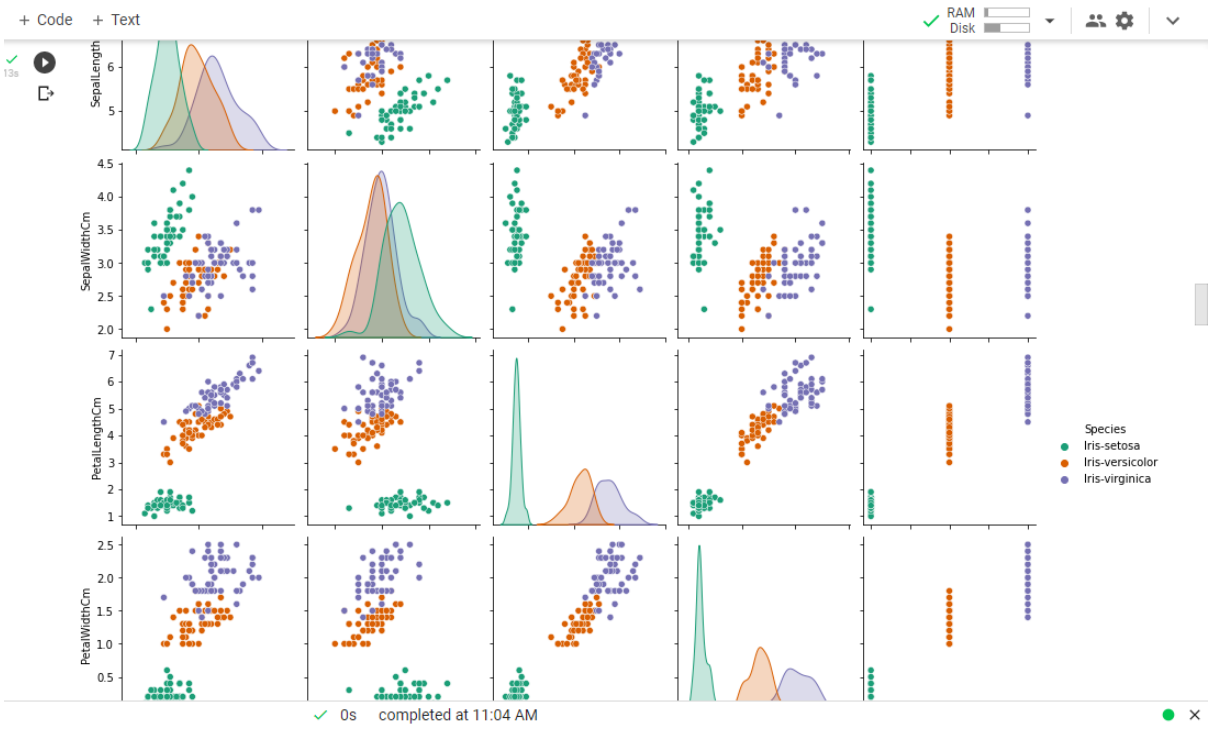



```
<Figure size 432x288 with 0 Axes>
```

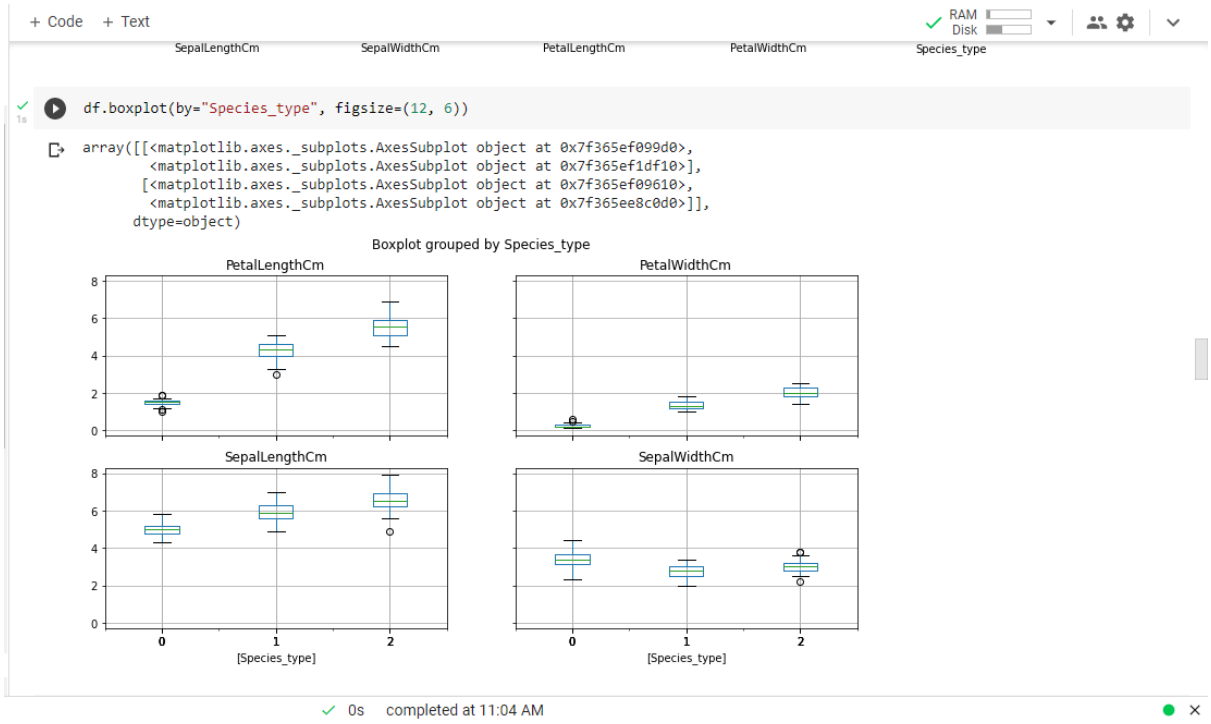
```
[14] sns.pairplot(df,hue='Species',palette='Dark2')
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:306: UserWarning: Dataset has 0 variance; skipping density estimate.
```

0s completed at 11:04 AM



0s completed at 11:04 AM




```
df.boxplot(by="Species_type", figsize=(12, 6))
```

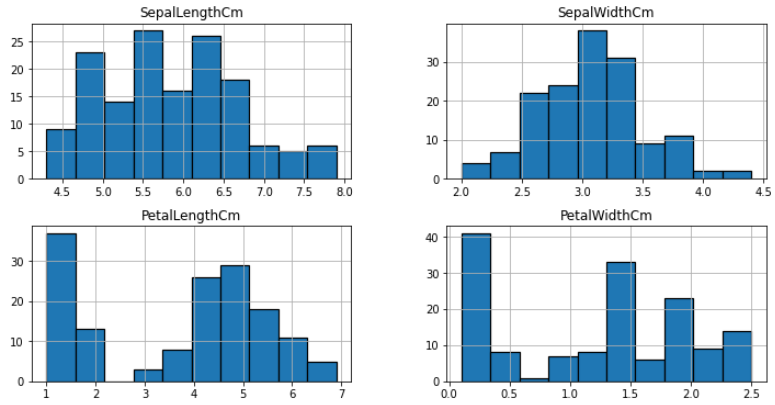
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7f365ef099d0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f365ef1df10>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7f365ef09610>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f365ee8c0d0>]],
      dtype=object)
```


0s completed at 11:04 AM

+ Code + Text

RAM Disk

```
[16] df.drop('Species_type',axis=1).hist(edgecolor='black', linewidth=1.2)
     fig=plt.gcf()
     fig.set_size_inches(12,6)
     plt.show()
```




```
[17] X_train,X_test,Y_train,Y_test = train_test_split(df.drop(['Species_type','Species'],axis=1),df['Species_type'],test_size=0.25,rando
```

```
[18] plt.figure(figsize=(8,5))
     sns.countplot(x=Y_test)
```

0s completed at 11:04 AM

+ Code + Text

RAM Disk

```
[19] models = ['random_forest','svm','decision_tree','logistic_regression','knn']
     model_train_acc=[]
     model_test_acc=[]
```

```
[20] svm= SVC()
     svm.fit(X_train,Y_train)
     y_hat_train = svm.predict(X_train)
     y_hat_test = svm.predict(X_test)
     train_acc = np.round(accuracy_score(y_hat_train,Y_train),3)
     test_acc =  np.round(accuracy_score(y_hat_test,Y_test),3)

     model_train_acc.append(train_acc)
     model_test_acc.append(test_acc)

     print("Accuracy Score on train data using SVM: ",train_acc)
     print("Accuracy Score on test data using SVM: ",test_acc)
```

```
Accuracy Score on train data using SVM:  0.955
Accuracy Score on test data using SVM:  0.947
```

```
[21] rfc = RandomForestClassifier()
     rfc.fit(X_train,Y_train)
     y_hat_train = rfc.predict(X_train)
     y_hat_test = rfc.predict(X_test)
     train_acc = np.round(accuracy_score(y_hat_train,Y_train),3)
     test_acc =  np.round(accuracy_score(y_hat_test,Y_test),3)

     model_train_acc.append(train_acc)
     model_test_acc.append(test_acc)
```

0s completed at 11:04 AM

+ Code + Text

RAM Disk

```
[21] print("Accuracy Score on train data using RandomForest: ",train_acc)
     print("Accuracy Score on test data using RandomForest: ",test_acc)
```

```
Accuracy Score on train data using RandomForest:  1.0
Accuracy Score on test data using RandomForest:  0.895
```

```
[22] tree = DecisionTreeClassifier()
     tree.fit(X_train,Y_train)
     y_hat_train = tree.predict(X_train)
     y_hat_test = tree.predict(X_test)
     train_acc = np.round(accuracy_score(y_hat_train,Y_train),3)
     test_acc =  np.round(accuracy_score(y_hat_test,Y_test),3)

     model_train_acc.append(train_acc)
     model_test_acc.append(test_acc)

     print("Accuracy Score on train data using Decision tree: ",train_acc)
     print("Accuracy Score on test data using Decision tree: ",test_acc)
```

```
Accuracy Score on train data using Decision tree:  1.0
Accuracy Score on test data using Decision tree:  0.895
```

```
[23] lr = LogisticRegression(max_iter=200)
     lr.fit(X_train,Y_train)
     y_hat_train = lr.predict(X_train)
     y_hat_test = lr.predict(X_test)
     train_acc = np.round(accuracy_score(y_hat_train,Y_train),3)
     test_acc =  np.round(accuracy_score(y_hat_test,Y_test),3)

     model_train_acc.append(train_acc)
```

✓ 0s completed at 11:04 AM

+ Code + Text

RAM Disk

```
[23] print("Accuracy Score on train data using Logistic Regression: ",train_acc)
     print("Accuracy Score on test data using Logistic Regression: ",test_acc)
```

```
Accuracy Score on train data using Logistic Regression:  0.973
Accuracy Score on test data using Logistic Regression:  0.921
```

```
[24] def select_neighbors():
         knn_train_acc=[]
         knn_test_acc=[]
         for i in range(1,11):
             knn = KNeighborsClassifier(n_neighbors=i)
             knn.fit(X_train,Y_train)
             y_hat_train = knn.predict(X_train)
             y_hat_test = knn.predict(X_test)
             knn_train_acc.append(accuracy_score(y_hat_train,Y_train))
             knn_test_acc.append(accuracy_score(y_hat_test,Y_test))

         return knn_train_acc,knn_test_acc
```

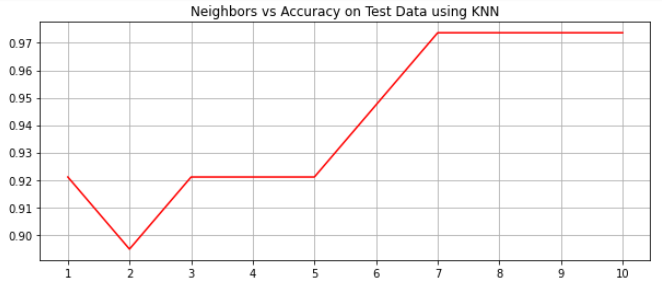
```
[25] knn_train,knn_test = select_neighbors()
     x = np.linspace(1,10,10)

     plt.figure(figsize=(10,4))
     plt.plot(x,knn_test,color='red')
     plt.title("Neighbors vs Accuracy on Test Data using KNN")
     plt.xticks(x)
     plt.grid()

     plt.figure(figsize=(10,4))
     plt.plot(x,knn_train,color='red')
     plt.title("Neighbors vs Accuracy on Train Data using KNN")
```

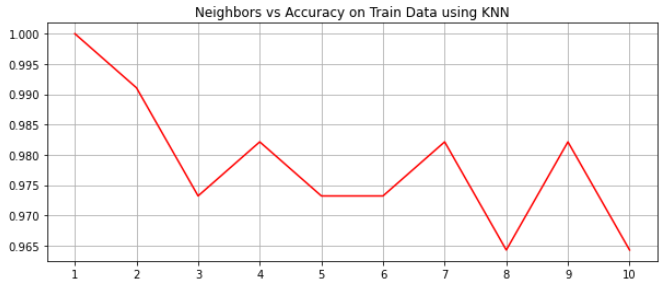
✓ 0s completed at 11:04 AM

+ Code + Text

RAM Disk

[25]






```
[26] train_acc = np.round(knn_train[6],3)
     test_acc = np.round(knn_test[6],3)
```

0s completed at 11:04 AM

+ Code + Text

RAM Disk

```
[26] train_acc = np.round(knn_train[6],3)
     test_acc = np.round(knn_test[6],3)

     model_train_acc.append(train_acc)
     model_test_acc.append(test_acc)

     print("Accuracy Score on train data using KNN: ",train_acc)
     print("Accuracy Score on test data using KNN: ",test_acc)
```

```
Accuracy Score on train data using KNN:  0.982
Accuracy Score on test data using KNN:  0.974
```

```
[ ]
```

0s completed at 11:04 AM