Data Science with Python Module 10

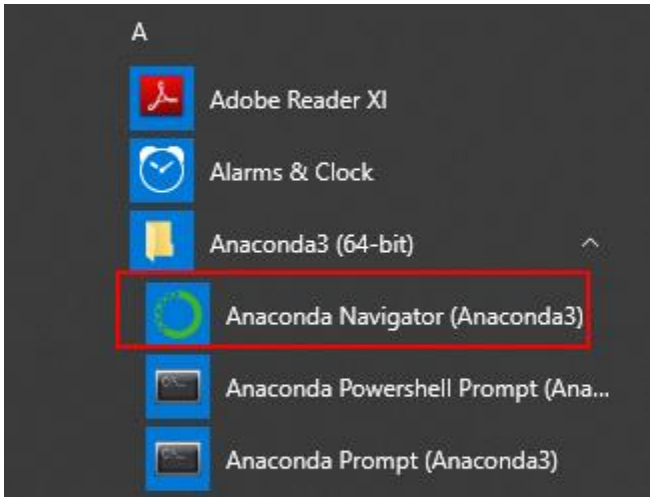
Hands On - 1

support@intellipaat.com
+91-7022374614
US: 1-800-216-8930(Toll Free)

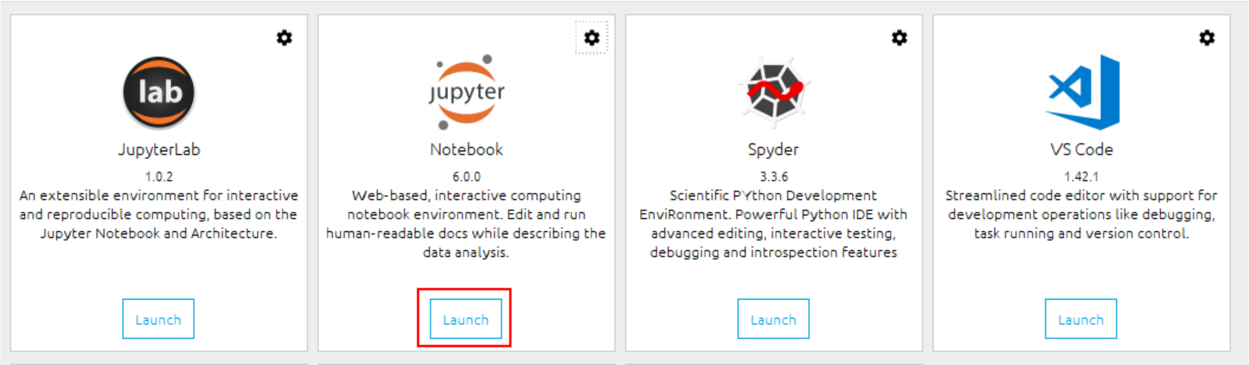
Data Science with Python Module 10: Hands-on: 1

Principal Component Analysis

Step 1: Open Anaconda Navigator


A
Adobe Reader XI
Alarms & Clock
Anaconda3 (64-bit)
Anaconda Navigator (Anaconda3)
Anaconda Powershell Prompt (Ana...
Anaconda Prompt (Anaconda3)

Step 2: Click on Launch button under jupyter notebooks.



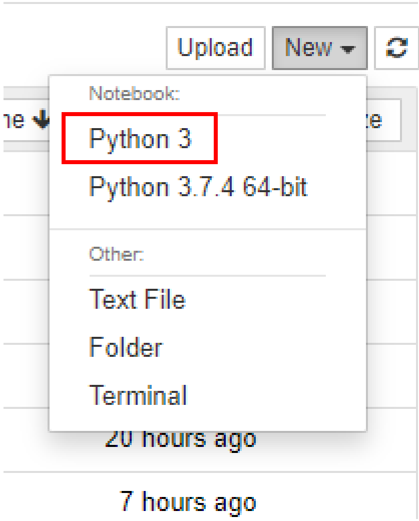
An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.

Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.

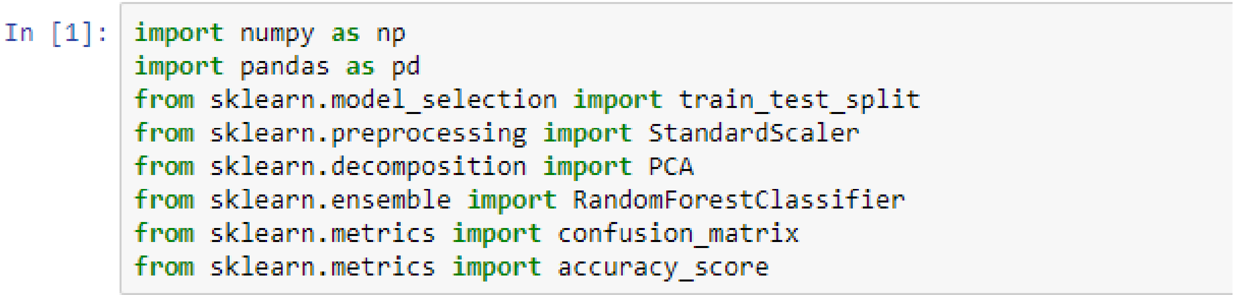
Scientific PYthon Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features

Streamlined code editor with support for development operations like debugging, task running and version control.

Step 3: After the notebook opens click on new and Python 3.

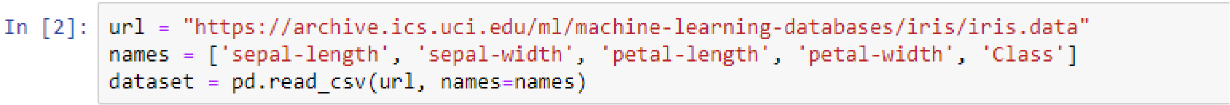


Step 4: Import all the required modules by typing the following code in the notebook and run it by pressing shift + enter



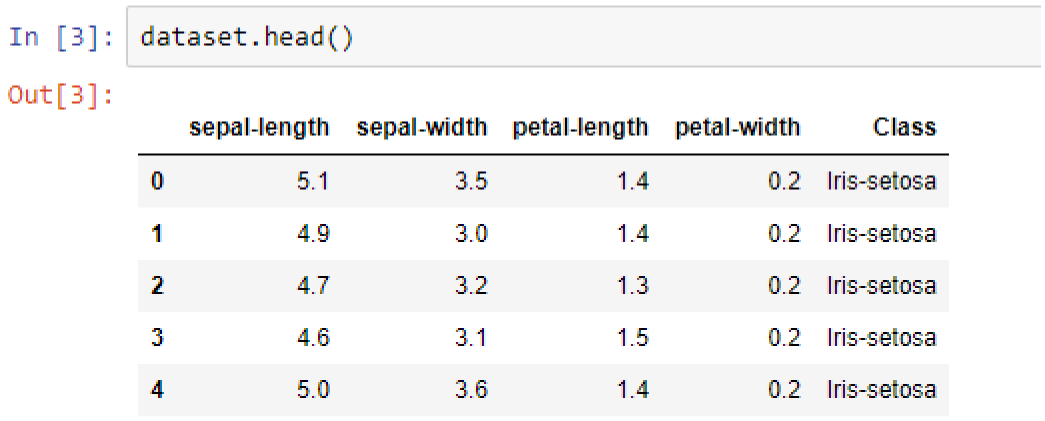
```
In [1]: import numpy as np
        import pandas as pd
        from sklearn.model_selection import train_test_split
        from sklearn.preprocessing import StandardScaler
        from sklearn.decomposition import PCA
        from sklearn.ensemble import RandomForestClassifier
        from sklearn.metrics import confusion_matrix
        from sklearn.metrics import accuracy_score
```

Step 5: Load the iris dataset.



```
In [2]: url = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"
        names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'Class']
        dataset = pd.read_csv(url, names=names)
```

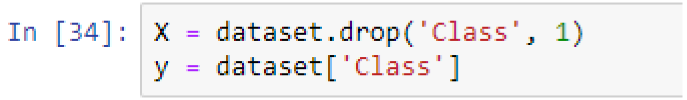
Step 6: Analyze the head of the data.



```
In [3]: dataset.head()
Out[3]:
```

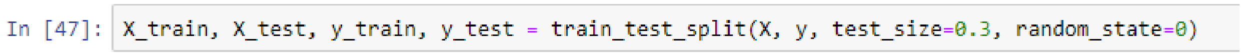
	sepal-length	sepal-width	petal-length	petal-width	Class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

Step 7: Extract data from dataframe into X and Y variables.



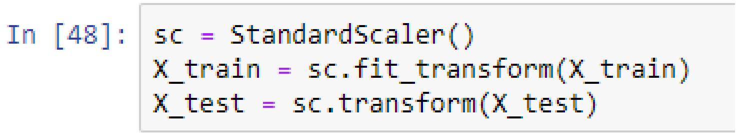
```
In [34]: X = dataset.drop('Class', 1)
         y = dataset['Class']
```

Step 8: Split the data into 70 percent for training and 30 percent testing.



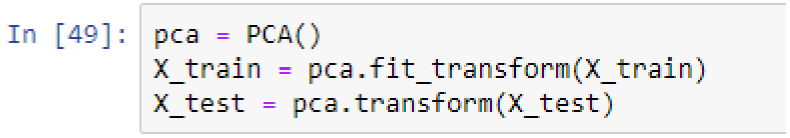
```
In [47]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)
```

Step 9: Scale the data.



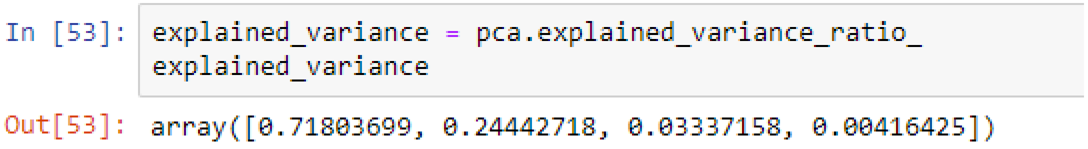
```
In [48]: sc = StandardScaler()
         X_train = sc.fit_transform(X_train)
         X_test = sc.transform(X_test)
```

Step 9: Create a PCA object and transform x_train and x_test.



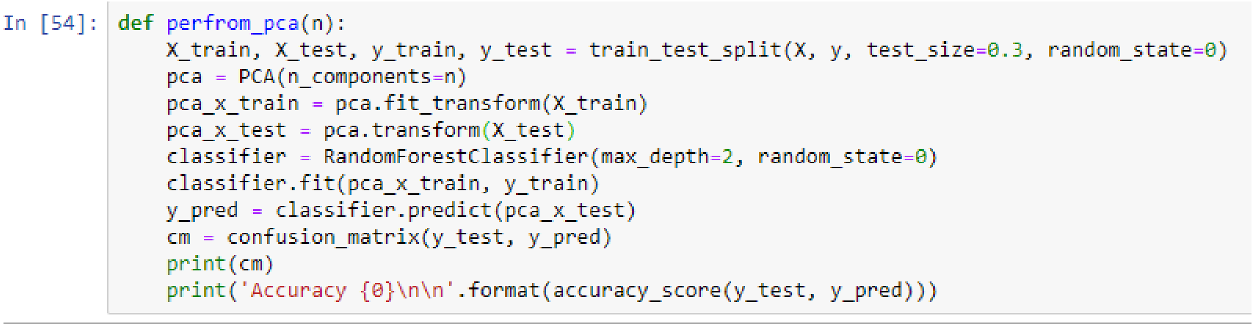
```
In [49]: pca = PCA()
         X_train = pca.fit_transform(X_train)
         X_test = pca.transform(X_test)
```

Step 10: Take a look at variance explained by each principal component.



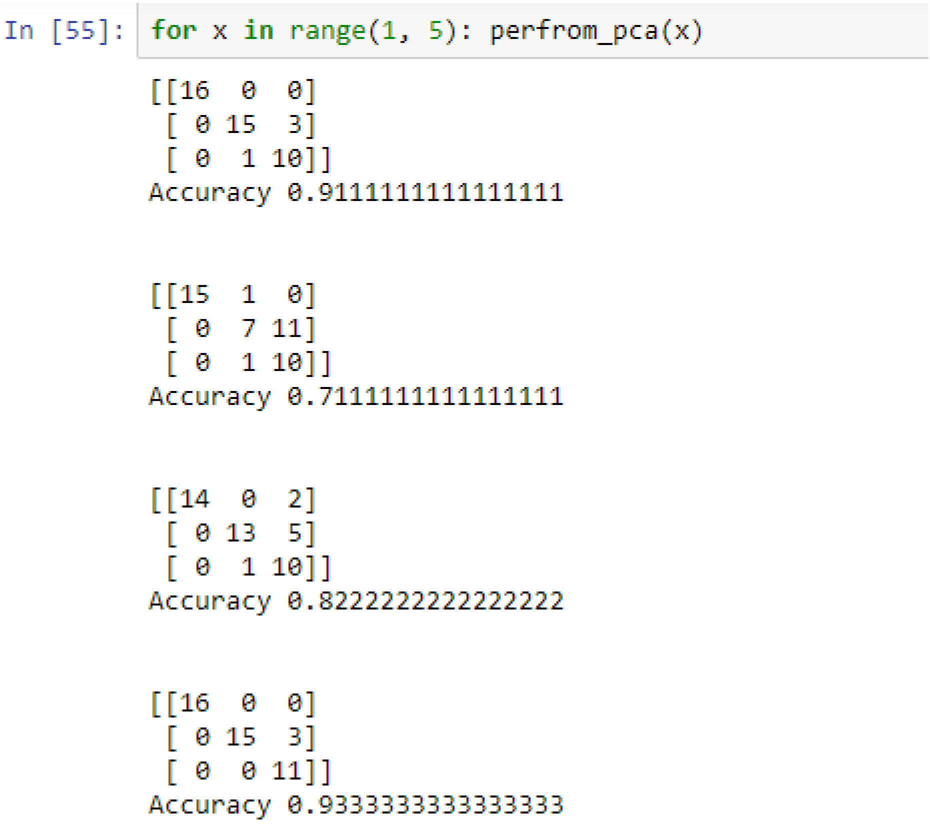
```
In [53]: explained_variance = pca.explained_variance_ratio_
         explained_variance

Out[53]: array([0.71803699, 0.24442718, 0.03337158, 0.00416425])
```

Step 11: Define a function called perform_pca that takes number of components for PCA to find and creates a RandomForestClassifier and calculates its accuracy.



```
In [54]: def perfrom_pca(n):
             X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)
             pca = PCA(n_components=n)
             pca_x_train = pca.fit_transform(X_train)
             pca_x_test = pca.transform(X_test)
             classifier = RandomForestClassifier(max_depth=2, random_state=0)
             classifier.fit(pca_x_train, y_train)
             y_pred = classifier.predict(pca_x_test)
             cm = confusion_matrix(y_test, y_pred)
             print(cm)
             print('Accuracy {0}\n\n'.format(accuracy_score(y_test, y_pred)))
```

Step 12: Call the perform_pca method with n_components set to a number from 1 to 4 and print their confusion matrix and accuracy scores.



```
In [55]: for x in range(1, 5): perfrom_pca(x)

         [[16  0  0]
          [ 0 15  3]
          [ 0  1 10]]
         Accuracy 0.9111111111111111


         [[15  1  0]
          [ 0  7 11]
          [ 0  1 10]]
         Accuracy 0.7111111111111111


         [[14  0  2]
          [ 0 13  5]
          [ 0  1 10]]
         Accuracy 0.8222222222222222


         [[16  0  0]
          [ 0 15  3]
          [ 0  0 11]]
         Accuracy 0.9333333333333333
```