Data Classification Analysis

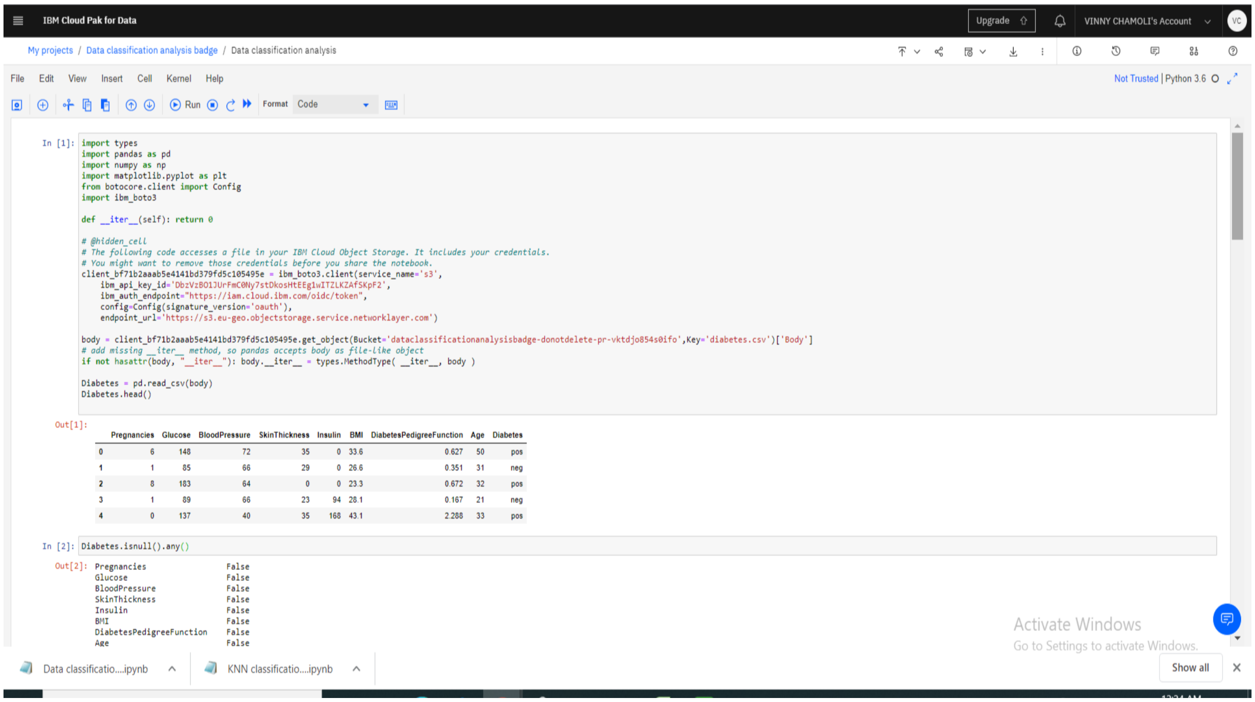
Diabetes Prediction Using Machine Learning

Logistic Regression

Logistic Regression is used when the dependent variable (target) is categorical.
For example,

- To predict whether an email is spam (1) or (0)
- Whether the tumor is malignant (1) or not (0)

Logistic regression **is used to describe data and to explain the relationship between one dependent binary variable and one or more nominal, ordinal, interval independent variables. In** Logistic Regression**, we don't directly fit a straight line to our data like in linear regression. Instead, we fit a S shaped curve, called** Sigmoid**, to our observations.**

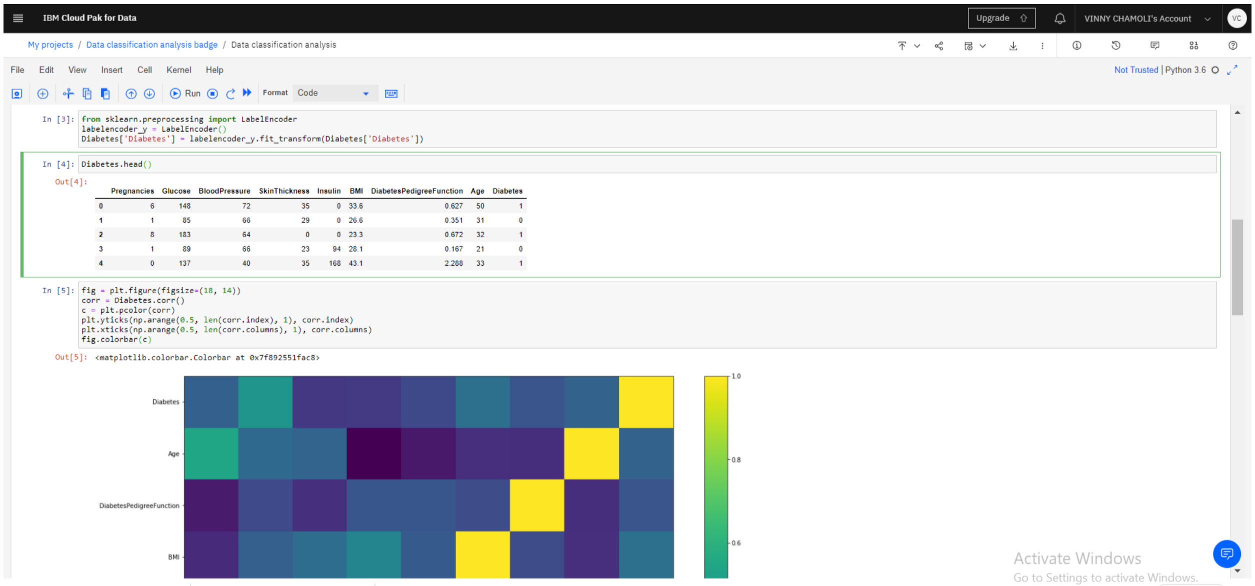

IBM Cloud Pak for Data
Upgrade
VINNY CHAMOLI's Account
VC
My projects / Data classification analysis badge / Data classification analysis
File Edit View Insert Cell Kernel Help
Not Trusted | Python 3.6
Run
Format Code
Activate Windows
Go to Settings to activate Windows.
Data classificatio....ipynb
KNN classificatio....ipynb
Show all

```
In [1]: import types
        import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        from botocore.client import Config
        import ibm_boto3

        def __iter__(self): return 0

        # @hidden_cell
        # The following code accesses a file in your IBM Cloud Object Storage. It includes your credentials.
        # You might want to remove those credentials before you share the notebook.
        client_bf71b2aaab5e4141bd379fd5c105495e = ibm_boto3.client(service_name='s3',
            ibm_api_key_id='DbzVzBO1JUrFmC0Ny7stDkosHtEEg1wITZLKZAfSKpF2',
            ibm_auth_endpoint="https://iam.cloud.ibm.com/oidc/token",
            config=Config(signature_version='oauth'),
            endpoint_url='https://s3.eu-geo.objectstorage.service.networklayer.com')

        body = client_bf71b2aaab5e4141bd379fd5c105495e.get_object(Bucket='dataclassificationanalysisbadge-donotdelete-pr-vktdjo854s0ifo',Key='diabetes.csv')['Body']
        # add missing __iter__ method, so pandas accepts body as file-like object
        if not hasattr(body, "__iter__"): body.__iter__ = types.MethodType( __iter__, body )

        Diabetes = pd.read_csv(body)
        Diabetes.head()
```

Out[1]:

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

```
In [2]: Diabetes.isnull().any()

Out[2]: Pregnancies                 False
        Glucose                     False
        BloodPressure               False
        SkinThickness               False
        Insulin                     False
        BMI                         False
        DiabetesPedigreeFunction    False
        Age                         False
```



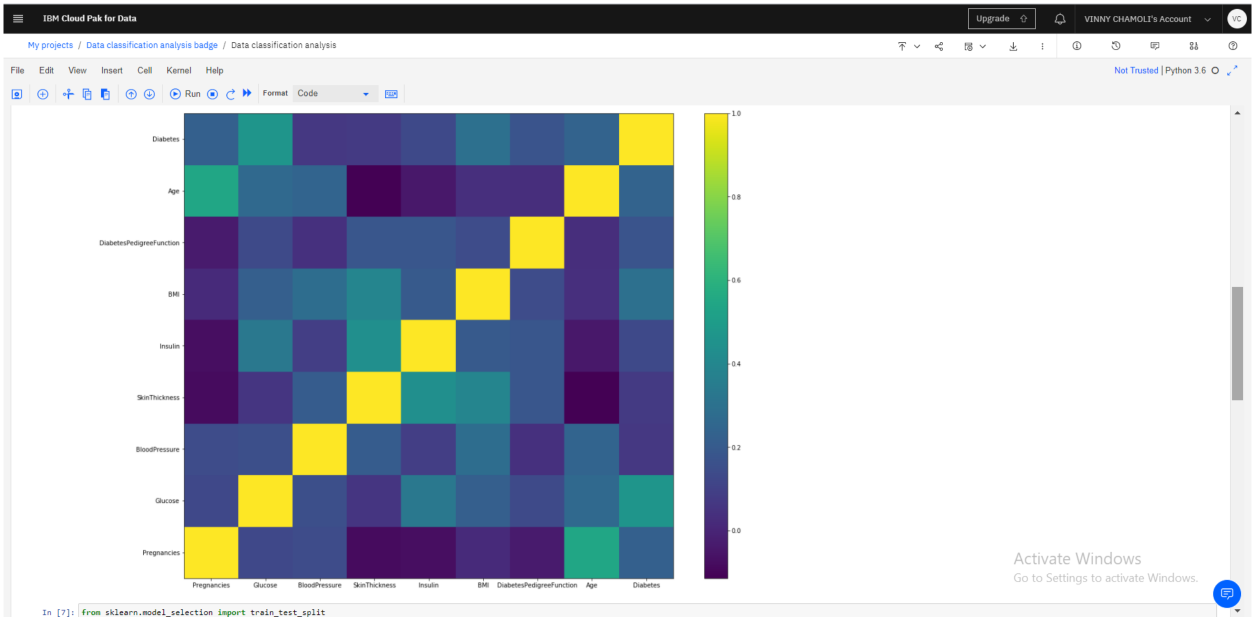
```
In [3]: from sklearn.preprocessing import LabelEncoder
        labelencoder_y = LabelEncoder()
        Diabetes['Diabetes'] = labelencoder_y.fit_transform(Diabetes['Diabetes'])
```

```
In [4]: Diabetes.head()
```

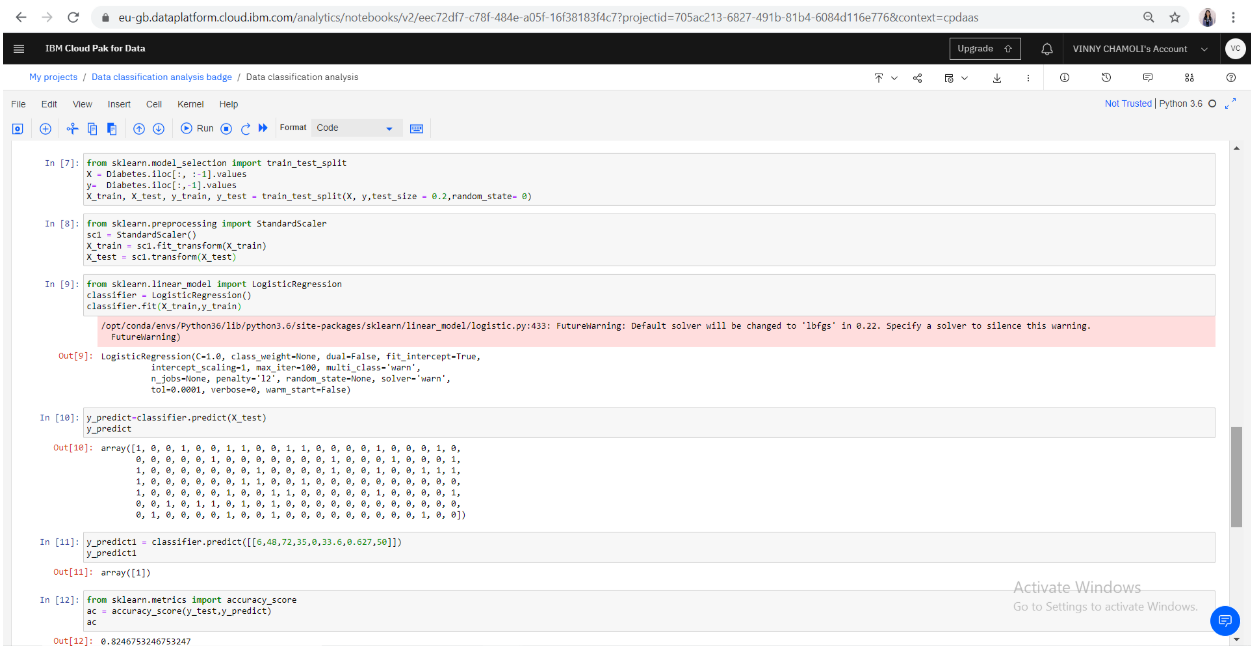
Out[4]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Diabetes
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

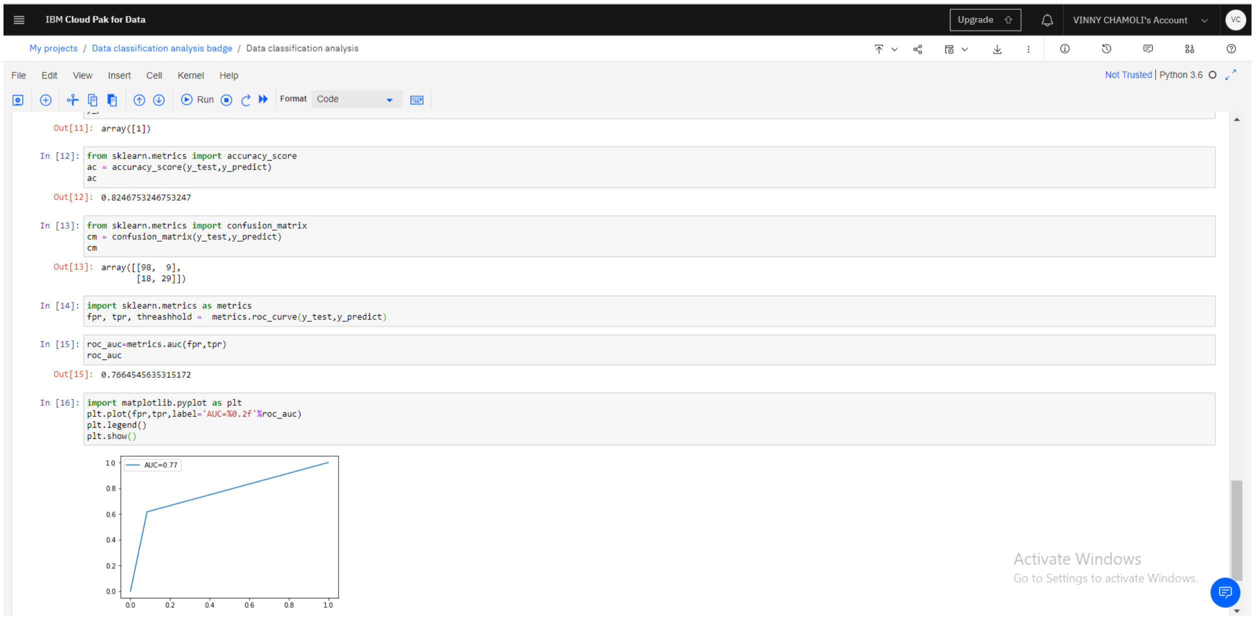
```
In [5]: fig = plt.figure(figsize=(18, 14))
        corr = Diabetes.corr()
        c = plt.pcolor(corr)
        plt.yticks(np.arange(0.5, len(corr.index), 1), corr.index)
        plt.xticks(np.arange(0.5, len(corr.columns), 1), corr.columns)
        fig.colorbar(c)

Out[5]: <matplotlib.colorbar.Colorbar at 0x7f892551fac8>
```



```
In [7]: from sklearn.model_selection import train_test_split
```


eu-gb.dataplatform.cloud.ibm.com/analytics/notebooks/v2/eec72df7-c78f-484e-a05f-16f38183f4c7?projectid=705ac213-6827-491b-81b4-6084d116e776&context=cpdaas
IBM Cloud Pak for Data
Upgrade
VINNY CHAMOLI's Account
VC
My projects / Data classification analysis badge / Data classification analysis
File
Edit
View
Insert
Cell
Kernel
Help
Not Trusted | Python 3.6
Run
Format
Code
Activate Windows
Go to Settings to activate Windows.

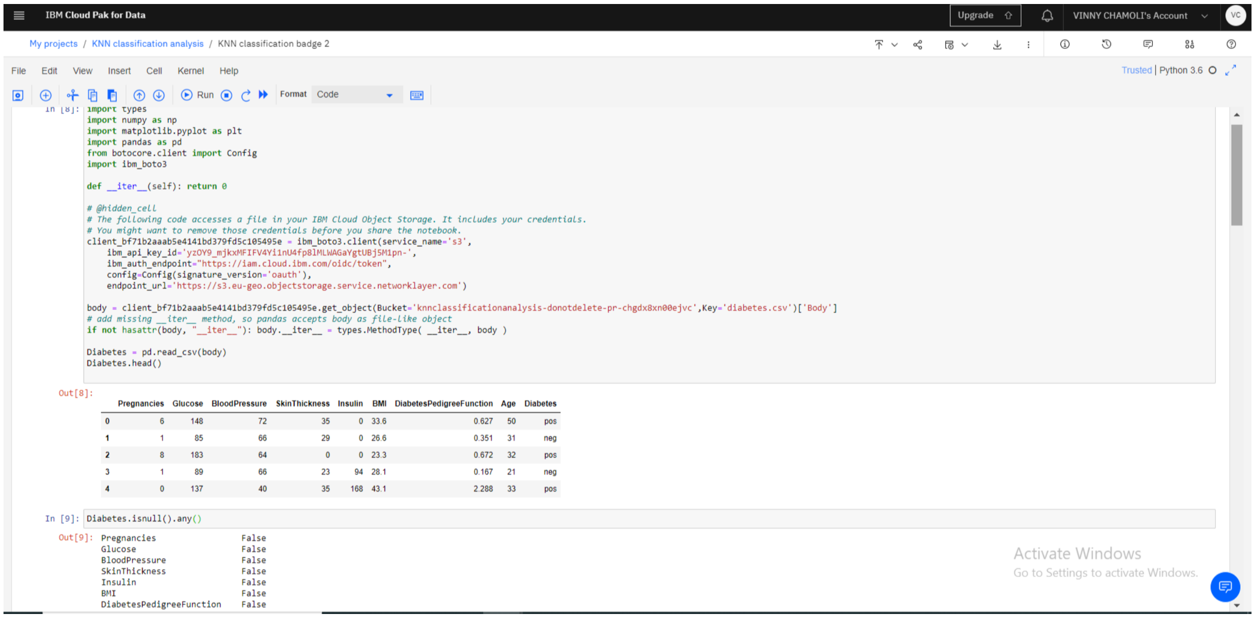
```
In [7]: from sklearn.model_selection import train_test_split
        X = Diabetes.iloc[:, :-1].values
        y=  Diabetes.iloc[:,-1].values
        X_train, X_test, y_train, y_test = train_test_split(X, y,test_size = 0.2,random_state= 0)

In [8]: from sklearn.preprocessing import StandardScaler
        sc1 = StandardScaler()
        X_train = sc1.fit_transform(X_train)
        X_test = sc1.transform(X_test)

In [9]: from sklearn.linear_model import LogisticRegression
        classifier = LogisticRegression()
        classifier.fit(X_train,y_train)

          /opt/conda/envs/Python36/lib/python3.6/site-packages/sklearn/linear_model/logistic.py:433: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.
            FutureWarning)

 Out[9]: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                    intercept_scaling=1, max_iter=100, multi_class='warn',
                    n_jobs=None, penalty='l2', random_state=None, solver='warn',
                    tol=0.0001, verbose=0, warm_start=False)

In [10]: y_predict=classifier.predict(X_test)
         y_predict

Out[10]: array([1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0,
                0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1,
                1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1,
                1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1,
                0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0])

In [11]: y_predict1 = classifier.predict([[6,48,72,35,0,33.6,0.627,50]])
         y_predict1

Out[11]: array([1])

In [12]: from sklearn.metrics import accuracy_score
         ac = accuracy_score(y_test,y_predict)
         ac

Out[12]: 0.8246753246753247
```



```
Out[11]: array([1])

In [12]: from sklearn.metrics import accuracy_score
         ac = accuracy_score(y_test,y_predict)
         ac

Out[12]: 0.8246753246753247

In [13]: from sklearn.metrics import confusion_matrix
         cm = confusion_matrix(y_test,y_predict)
         cm

Out[13]: array([[98,  9],
                [18, 29]])

In [14]: import sklearn.metrics as metrics
         fpr, tpr, threashhold =  metrics.roc_curve(y_test,y_predict)

In [15]: roc_auc=metrics.auc(fpr,tpr)
         roc_auc

Out[15]: 0.7664545635315172

In [16]: import matplotlib.pyplot as plt
         plt.plot(fpr,tpr,label='AUC=%0.2f'%roc_auc)
         plt.legend()
         plt.show()
```

K Nearest Neighbours Classification Algorithm

The k-nearest neighbors (KNN) algorithm is a simple, easy-to-implement supervised machine learning algorithm that can be used to solve both classification. KNN has no model other than storing the entire dataset, so there is no learning required.
KNN works by finding the distances between a query and all the examples in the data, selecting the specified number examples (K) closest to the query, then votes for the most frequent label (in the case of classification)
A case is classified by a majority vote of its neighbors, with the case being assigned to the class most common amongst its K nearest neighbors measured by a distance function. If K = 5, then the case is simply assigned to the class which has highest majority. It is a memory based algorithm. In this test time is greater than training time and more over K should be an odd number as to classify into either of the categories k should be an odd number. Feature Scaling is must for KNN as the algorithm uses distance calculation for its approach.

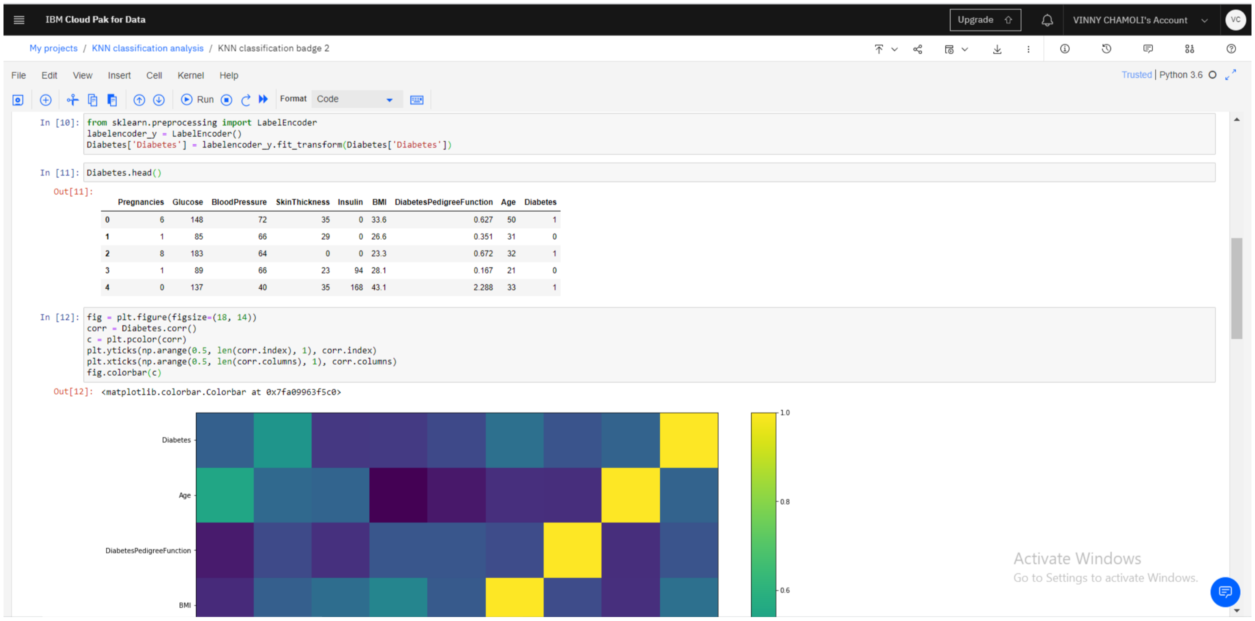

IBM Cloud Pak for Data
Upgrade
VINNY CHAMOLI's Account
VC
My projects / KNN classification analysis / KNN classification badge 2
File Edit View Insert Cell Kernel Help
Run
Format Code
Trusted | Python 3.6

```
In [8]: import types
        import numpy as np
        import matplotlib.pyplot as plt
        import pandas as pd
        from botocore.client import Config
        import ibm_boto3

        def __iter__(self): return 0

        # @hidden_cell
        # The following code accesses a file in your IBM Cloud Object Storage. It includes your credentials.
        # You might want to remove those credentials before you share the notebook.
        client_bf71b2aaab5e4141bd379fd5c105495e = ibm_boto3.client(service_name='s3',
            ibm_api_key_id='yzOY9_mjkxMFIFV4Yi1nU4fp8lMLWAGaYgtUBj5M1pn-',
            ibm_auth_endpoint="https://iam.cloud.ibm.com/oidc/token",
            config=Config(signature_version='oauth'),
            endpoint_url='https://s3.eu-geo.objectstorage.service.networklayer.com')

        body = client_bf71b2aaab5e4141bd379fd5c105495e.get_object(Bucket='knnclassificationanalysis-donotdelete-pr-chgdx8xn00ejvc',Key='diabetes.csv')['Body']
        # add missing __iter__ method, so pandas accepts body as file-like object
        if not hasattr(body, "__iter__"): body.__iter__ = types.MethodType( __iter__, body )

        Diabetes = pd.read_csv(body)
        Diabetes.head()
```

Out[8]:

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

```
In [9]: Diabetes.isnull().any()

Out[9]: Pregnancies                 False
        Glucose                     False
        BloodPressure               False
        SkinThickness               False
        Insulin                     False
        BMI                         False
        DiabetesPedigreeFunction    False
```



```
In [10]: from sklearn.preprocessing import LabelEncoder
         labelencoder_y = LabelEncoder()
         Diabetes['Diabetes'] = labelencoder_y.fit_transform(Diabetes['Diabetes'])
```

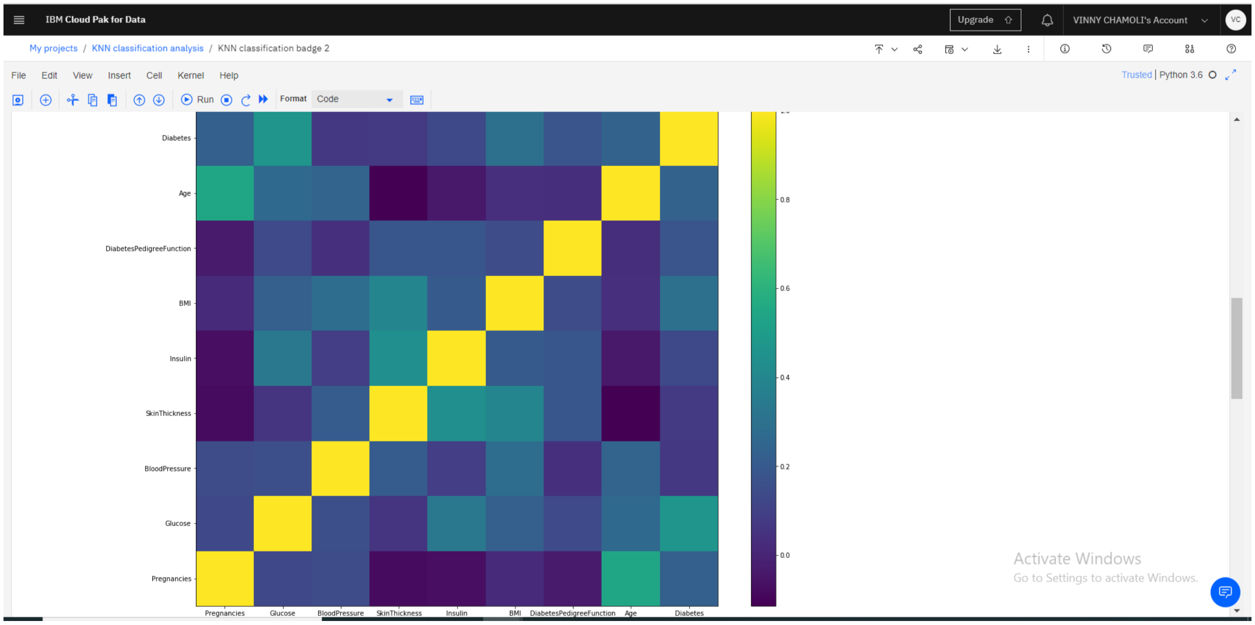
```
In [11]: Diabetes.head()
```

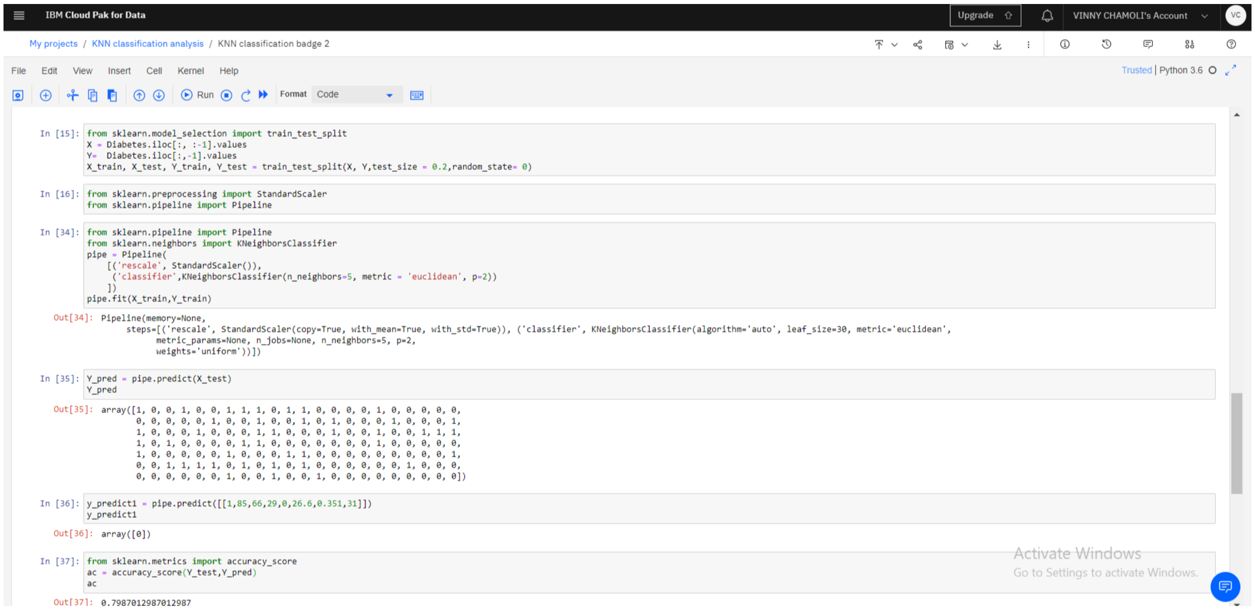
Out[11]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Diabetes
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

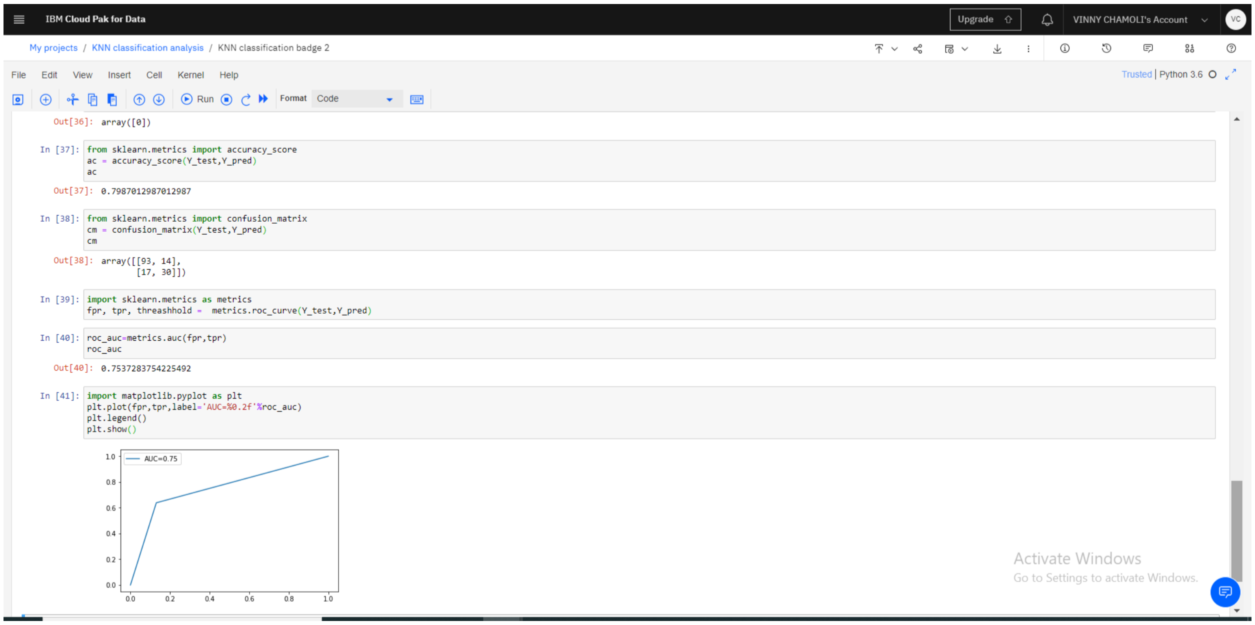
```
In [12]: fig = plt.figure(figsize=(18, 14))
         corr = Diabetes.corr()
         c = plt.pcolor(corr)
         plt.yticks(np.arange(0.5, len(corr.index), 1), corr.index)
         plt.xticks(np.arange(0.5, len(corr.columns), 1), corr.columns)
         fig.colorbar(c)
```

```
Out[12]: <matplotlib.colorbar.Colorbar at 0x7fa09963f5c0>
```





```
In [15]: from sklearn.model_selection import train_test_split
         X = Diabetes.iloc[:, :-1].values
         Y=  Diabetes.iloc[:,-1].values
         X_train, X_test, Y_train, Y_test = train_test_split(X, Y,test_size = 0.2,random_state= 0)

In [16]: from sklearn.preprocessing import StandardScaler
         from sklearn.pipeline import Pipeline

In [34]: from sklearn.pipeline import Pipeline
         from sklearn.neighbors import KNeighborsClassifier
         pipe = Pipeline(
             [('rescale', StandardScaler()),
              ('classifier',KNeighborsClassifier(n_neighbors=5, metric = 'euclidean', p=2))
             ])
         pipe.fit(X_train,Y_train)

Out[34]: Pipeline(memory=None,
             steps=[('rescale', StandardScaler(copy=True, with_mean=True, with_std=True)), ('classifier', KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='euclidean',
                    metric_params=None, n_jobs=None, n_neighbors=5, p=2,
                    weights='uniform'))])

In [35]: Y_pred = pipe.predict(X_test)
         Y_pred

Out[35]: array([1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1,
                1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1,
                1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
                1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
                0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0])

In [36]: y_predict1 = pipe.predict([[1,85,66,29,0,26.6,0.351,31]])
         y_predict1

Out[36]: array([0])

In [37]: from sklearn.metrics import accuracy_score
         ac = accuracy_score(Y_test,Y_pred)
         ac

Out[37]: 0.7987012987012987
```



```
Out[36]: array([0])

In [37]: from sklearn.metrics import accuracy_score
         ac = accuracy_score(Y_test,Y_pred)
         ac

Out[37]: 0.7987012987012987
```

```
In [38]: from sklearn.metrics import confusion_matrix
         cm = confusion_matrix(Y_test,Y_pred)
         cm

Out[38]: array([[93, 14],
                [17, 30]])

In [39]: import sklearn.metrics as metrics
         fpr, tpr, threashhold =  metrics.roc_curve(Y_test,Y_pred)

In [40]: roc_auc=metrics.auc(fpr,tpr)
         roc_auc

Out[40]: 0.7537283754225492

In [41]: import matplotlib.pyplot as plt
         plt.plot(fpr,tpr,label='AUC=%0.2f'%roc_auc)
         plt.legend()
         plt.show()
```