Welcome to Hex!

Published 2 min ago by RITHVIKK

Get started with this example project that uses SQL and Python to find the most popular dessert order for a fictional dumpling restaurant.

```
# Pandas is a software library written for the Python programming language for data manipulation and analysis.
import pandas as pd
# NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along
import numpy as np
# Matplotlib is a plotting library for python and pyplot gives us a Matlab like plotting framework. We will use this in our plot
import matplotlib.pyplot as plt
#Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive a
import seaborn as sns
# Preprocessing allows us to standarsize our data
from sklearn import preprocessing
# Allows us to split our data into training and testing data
from sklearn.model_selection import train_test_split
# Allows us to test parameters of classification algorithms and find the best one
from sklearn.model_selection import GridSearchCV
# Logistic Regression classification algorithm
from sklearn.linear_model import LogisticRegression
# Support Vector Machine classification algorithm
from sklearn.svm import SVC
# Decision Tree classification algorithm
from sklearn.tree import DecisionTreeClassifier
# K Nearest Neighbors classification algorithm
from sklearn.neighbors import KNeighborsClassifier
```

```
def plot_confusion_matrix(y,y_predict):
    "this function plots the confusion matrix"
    from sklearn.metrics import confusion_matrix

    cm = confusion_matrix(y, y_predict)
    ax= plt.subplot()
    sns.heatmap(cm, annot=True, ax = ax); #annot=True to annotate cells
    ax.set_xlabel('Predicted labels')
    ax.set_ylabel('True labels')
    ax.set_title('Confusion Matrix');
    ax.xaxis.set_ticklabels(['did not land', 'land']); ax.yaxis.set_ticklabels(['did not land', 'land'])
    plt.show()
```

```
import requests
import pandas as pd
import io

# URL of the CSV file

URL = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/dataset_part_2.csv
# Fetching the CSV file
response = requests.get(URL)
# Ensure the request was successful
if response.status_code == 200:
    # Read the CSV file into a pandas DataFrame
    data= pd.read_csv(io.StringIO(response.text))
    print('Data read into a pandas dataframe!')
    # Display the first few rows of the DataFrame
    print(data.head())
else:
    print(f"Failed to fetch the CSV file. Status code: {response.status_code}")
```

```
Data read into a pandas dataframe!
   FlightNumber        Date BoosterVersion  PayloadMass Orbit    LaunchSite  \
0             1  2010-06-04       Falcon 9  6104.959412   LEO  CCAFS SLC 40
1             2  2012-05-22       Falcon 9   525.000000   LEO  CCAFS SLC 40
2             3  2013-03-01       Falcon 9   677.000000   ISS  CCAFS SLC 40
3             4  2013-09-29       Falcon 9   500.000000    PO   VAFB SLC 4E
4             5  2013-12-03       Falcon 9  3170.000000   GTO  CCAFS SLC 40

      Outcome  Flights  GridFins  Reused   Legs LandingPad  Block  \
0    None None        1     False   False  False        NaN    1.0
1    None None        1     False   False  False        NaN    1.0
2    None None        1     False   False  False        NaN    1.0
3   False Ocean       1     False   False  False        NaN    1.0
4    None None        1     False   False  False        NaN    1.0

   ReusedCount Serial    Longitude   Latitude  Class
0            0  B0003  -80.577366  28.561857      0
1            0  B0005  -80.577366  28.561857      0
2            0  B0007  -80.577366  28.561857      0
3            0  B1003 -120.610829  34.632093      0
4            0  B1004  -80.577366  28.561857      0
```

```
Y=data['Class'].to_numpy()
```

```
from sklearn import preprocessing

# Assuming X is already defined and contains your data
transform = preprocessing.StandardScaler()

# Fit the StandardScaler to the data
transform.fit(X)

# Transform the data and assssign it to X
X = transform.transform(X)
```

```
X_train, X_test, Y_train, Y_test=train_test_split(X,Y,random_state=2,test_size=0.2)
```

```
(18,)
```

```
parametres ={'C':[0.01,0.1,1],'penalty':['l2'], 'solver':['lbfgs']}# 11 lasso 12 ridge
lr=LogisticRegression()
logreg_cv = GridSearchCV(estimator=lr, param_grid=parameters, cv=10).fit(X_train,Y_train)
```

```
print("tuned hpyerparameters :(best parameters) ",logreg_cv.best_params_)
print("accuracy :",logreg_cv.best_score_)
```

```
tuned hpyerparameters :(best parameters)  {'C': 0.01, 'penalty': 'l2', 'solver': 'lbfgs'}
accuracy : 0.8464285714285713
```

```
accuracy = logreg_cv.best_estimator_.score(X_test, Y_test)
accuracy
```

```
0.8333333333333334
```

```
yhat=logreg_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```


```
parametres1 = {'kernel':('linear', 'rbf','poly','rbf', 'sigmoid'),
              'C': np.logspace(-3, 3, 5),
              'gamma':np.logspace(-3, 3, 5)}

svm = SVC()
svm_cv = GridSearchCV(estimator=svm, param_grid=parametres1, cv=10).fit(X_train,Y_train)
print("tuned hpyerparameters :(best parameters) ",svm_cv.best_params_)
print( "accuracy :",svm_cv.best_score_)
```

```
tuned hpyerparameters :(best parameters)  {'C': 1.0, 'gamma': 0.03162277660168379, 'kernel': 'sigmoid'}
accuracy : 0.8482142857142856
```

```
accuracy1 = svm_cv.best_estimator_.score(X_test, Y_test)
accuracy1
```

```
0.8333333333333334
```

```
yhat=svm_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```


```
parameters = {'criterion': ['gini', 'entropy'],
     'splitter': ['best', 'random'],
     'max_depth': [2*n for n in range(1,10)],
     'max_features': ['auto', 'sqrt'],
     'min_samples_leaf': [1, 2, 4],
     'min_samples_split': [2, 5, 10]}

tree = DecisionTreeClassifier()
tree_cv = GridSearchCV(estimator=tree, param_grid=parameters, cv=10).fit(X_train,Y_train)
```

> Cannot display "text/plain" output. Size of 969 KB is larger than the 512 KB limit.

```
print("tuned hpyerparameters :(best parameters) ",tree_cv.best_params_)
print("accuracy :",tree_cv.best_score_)
```

```
tuned hpyerparameters :(best parameters)  {'criterion': 'entropy', 'max_depth': 6, 'max_features': 'auto', 'min_samples_leaf
accuracy : 0.9035714285714287
```

```
accuracy2 = tree_cv.best_estimator_.score(X_test, Y_test)
accuracy2
yhat = tree_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```


```
parameters = {'n_neighbors': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
              'algorithm': ['auto', 'ball_tree', 'kd_tree', 'brute'],
              'p': [1,2]}

KNN = KNeighborsClassifier()
knn_cv = GridSearchCV(estimator=KNN, param_grid=parameters, cv=10).fit(X_train,Y_train)
print("tuned hpyerparameters :(best parameters) ",knn_cv.best_params_)
print("accuracy :",knn_cv.best_score_)
```

```
tuned hpyerparameters :(best parameters)  {'algorithm': 'auto', 'n_neighbors': 10, 'p': 1}
accuracy : 0.8482142857142858
```

```
accuracy3 = knn_cv.best_estimator_.score(X_test, Y_test)
accuracy3
yhat = knn_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```


```
report = pd.DataFrame(
    {
        "names": ['logistic', 'tree', 'svm', 'knn'],
        "values": [logreg_cv.best_score_,tree_cv.best_score_ ,svm_cv.best_score_,knn_cv.best_score_],
    }
)
print(report)
```

```
    names    values
0  logistic  0.846429
1      tree  0.903571
2       svm  0.848214
3       knn  0.848214
```

```
report
```

	names	values
0	logistic	0.846429
1	tree	0.903571
2	svm	0.848214
3	knn	0.848214

4 rows

```
plt.bar(x=report["names"], height=report["values"], color="red")
plt.show()
```