da-lab-7

April 22, 2023

[54]:
```
#importing packages
import pandas as pd
from pandas import read_csv
from sklearn.model_selection import train_test_split
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import f_classif
from matplotlib import pyplot
from sklearn.feature_selection import mutual_info_classif
#Building model with all features
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
```

[55]:
```
data = read_csv("/content/drive/MyDrive/Project/amazon(DA).csv")
data.head(5)
```

[55]:

	discounted_price	actual_price	discount_percentage	rating	rating_count
0	399.0	1099.0	8755.0	4.2	24269.0
1	199.0	349.0	4300.0	4.0	43994.0
2	199.0	1899.0	9000.0	3.9	7928.0
3	329.0	699.0	5300.0	4.2	94363.0
4	154.0	399.0	6100.0	4.2	16905.0

[56]:
```
#loading dataset
def load_dataset(fname):
 data.dropna(inplace=True)
 precision=3
# data['actual_price'] = data['actual_price'].apply(lambda x: x.replace(',',␣
 ↪'')).astype('float64')
# data['discounted_price'] = data['discounted_price'].apply(lambda x: x.
 ↪replace(',', '')).astype('float64')
# data['rating_count'] = data['rating_count'].apply(lambda x: str(x).
 ↪replace(',', '0')).astype('float64')
# data['rating_count'] = data['rating_count'].apply(lambda x: str(x).
 ↪replace('|', '0')).astype('float64')
# data['rating_count'] = data['rating_count'].apply(lambda x: round(x,␣
 ↪precision))
 dataset=data.values
```

```
 X=dataset[:,:-1]
 y=dataset[:,-1]
 return X,y
```

[82]:
```
data.head(100)
```

[82]:

	discounted_price	actual_price	discount_percentage	rating	rating_count
0	399.0	1099.0	8755.0	4.2	24269.0
1	199.0	349.0	4300.0	4.0	43994.0
2	199.0	1899.0	9000.0	3.9	7928.0
3	329.0	699.0	5300.0	4.2	94363.0
4	154.0	399.0	6100.0	4.2	16905.0
..	…	…	…	…	…
95	290.0	349.0	1700.0	3.7	1977.0
96	249.0	799.0	6900.0	3.8	1079.0
97	345.0	999.0	6500.0	3.7	1097.0
98	1099.0	1899.0	4200.0	4.5	22420.0
99	719.0	1499.0	5200.0	4.1	1045.0

[100 rows x 5 columns]

[84]:
```
#Train Test split
X,y=load_dataset('/content/drive/MyDrive/Project/amazon(DA).csv')
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.
 ↪33,random_state=1)
print('Train',X_train.shape,y_train.shape)
print('Test',X_test.shape,y_test.shape)
```

```
Train (980, 4) (980,)
Test (483, 4) (483,)
```

[85]:
```
y.shape()
```

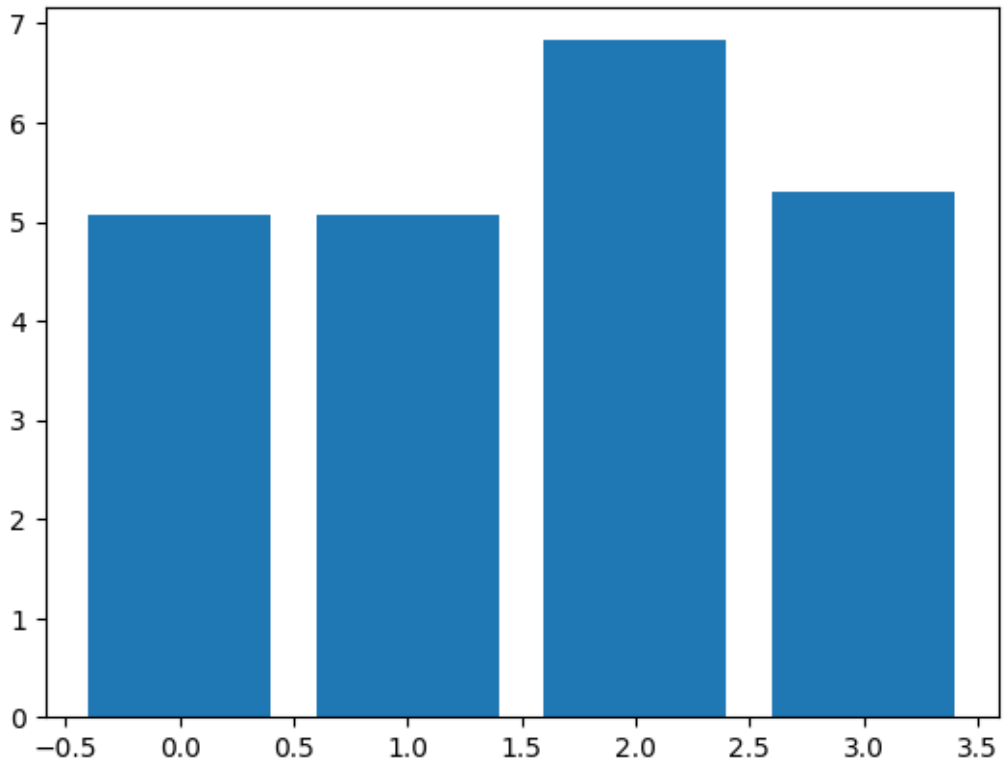
```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-85-c5cc93b6f9c2> in <cell line: 1>()
----> 1 y.shape()

TypeError: 'tuple' object is not callable
```

[59]:
```
#ANOVA F_statistic score based feature selection
def select_features(X_train,y_train,X_test):
 fs=SelectKBest(score_func=f_classif,k='all')
 fs.fit(X_train,y_train)
 X_train_fs=fs.transform(X_train)
 X_test_fs=fs.transform(X_test)
```

```
 return X_train_fs,X_test_fs,fs
```

[60]:
```
#Selecting best features and plotting
X_train_fs,X_test_fs,fs=select_features(X_train,y_train,X_test)
pyplot.bar([i for i in range(len(fs.scores_))],fs.scores_)
pyplot.show()
```



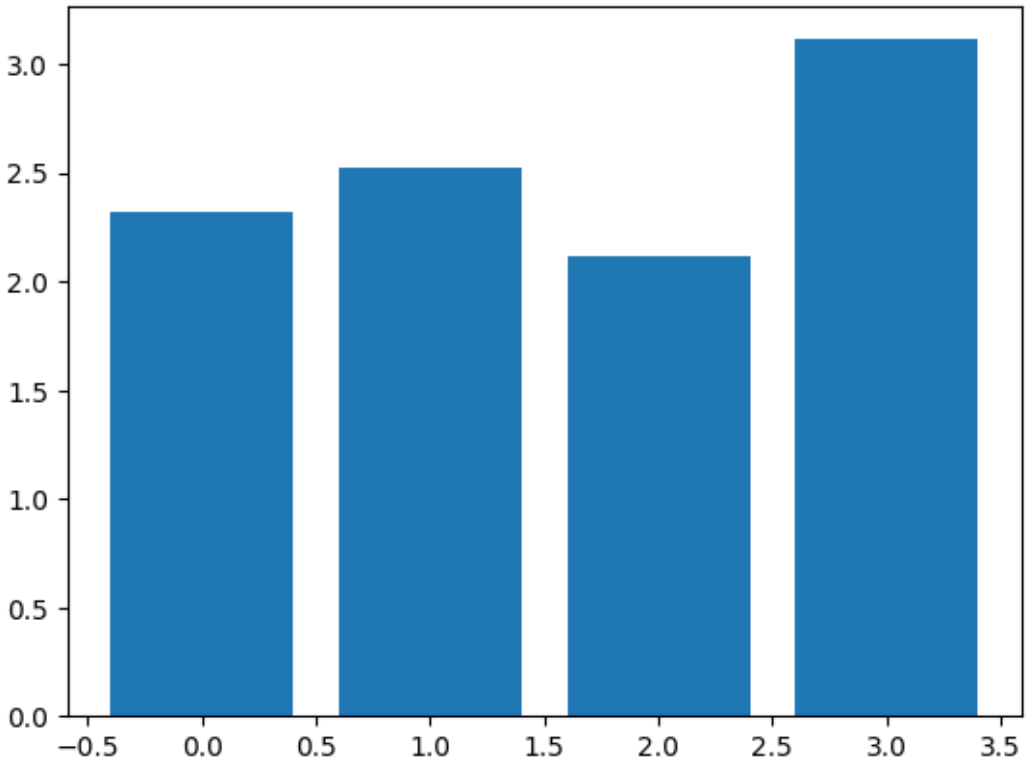
[61]:
```
#feature scores
fs.scores_
```

[61]:
```
array([5.06763542, 5.0782702 , 6.82967654, 5.30359687])
```

[62]:
```
#Mutual information based feature selection
def select_features_2(X_train,y_train,X_test):
 fs=SelectKBest(score_func=mutual_info_classif,k=4)
 fs.fit(X_train,y_train)
 X_train_fs=fs.transform(X_train)
 X_test_fs=fs.transform(X_test)
 return X_train_fs,X_test_fs,fs
```

[63]:
```
#Selecting best features using MI and plotting
X_train_fs2,X_test_fs2,fs2=select_features_2(X_train,y_train,X_test)
```

```
pyplot.bar([i for i in range(len(fs2.scores_))],fs2.scores_)
pyplot.show()
```



[64]:
```
#feature scores
fs2.scores_
```

[64]: array([2.32001347, 2.52288031, 2.11860866, 3.11375566])

[65]:
```
model1=LogisticRegression(solver='liblinear')
model1.fit(X_train,y_train)
yhat=model1.predict(X_test)
accuracy=accuracy_score(y_test,yhat)
print("Accuracy: %.2f" %(accuracy*100))
```

Accuracy: 1.86

[66]:
```
#Model built using features chosen with ANOVA F-statistic
model2=LogisticRegression(solver='liblinear')
model2.fit(X_train_fs,y_train)
yhat=model2.predict(X_test_fs)
accuracy=accuracy_score(y_test,yhat)
print("Accuracy: %.2f" %(accuracy*100))
```

```
Accuracy: 1.86
```

[67]:
```
#Model built using features chosen with Mutual information
model3=LogisticRegression(solver='liblinear')
model3.fit(X_train_fs2,y_train)
yhat=model3.predict(X_test_fs2)
accuracy=accuracy_score(y_test,yhat)
print("Accuracy: %.2f" %(accuracy*100))
```

```
Accuracy: 1.86
```

[68]:
```
#Tune the number of selected features -grid search
from sklearn.pipeline import Pipeline
from sklearn.model_selection import RepeatedStratifiedKFold
from sklearn.model_selection import GridSearchCV
```

[69]:
```
#define the data set
X,y=load_dataset('/content/drive/MyDrive/Project/amazon(DA).csv')
```

[70]:
```
#define the evaluation method - k-fold cross validation & k=2
cv=RepeatedStratifiedKFold(n_splits=2,n_repeats=3,random_state=1)
```

[71]:
```
#define the pipeline to evaluate
model=LogisticRegression(solver='liblinear')
fs=SelectKBest(score_func=f_classif)
pipeline=Pipeline(steps=[('anova',fs),('lr',model)])
```

[72]:
```
#define the grid
grid=dict()
grid['anova__k']=[i+1 for i in range(X.shape[1])]
```

[73]:
```
#define the grid search
search=GridSearchCV(pipeline, grid, scoring='accuracy',n_jobs=-1, cv=cv)
results=search.fit(X,y)
```

```
/usr/local/lib/python3.9/dist-packages/sklearn/model_selection/_split.py:700:
UserWarning: The least populated class in y has only 1 members, which is less
than n_splits=2.
  warnings.warn(
/usr/local/lib/python3.9/dist-packages/sklearn/model_selection/_split.py:700:
UserWarning: The least populated class in y has only 1 members, which is less
than n_splits=2.
  warnings.warn(
/usr/local/lib/python3.9/dist-packages/sklearn/model_selection/_split.py:700:
UserWarning: The least populated class in y has only 1 members, which is less
than n_splits=2.
  warnings.warn(
```

[74]:
```
print('Best Mean Accuracy: %.3f' % results.best_score_)
print('Best config %s' %results.best_params_)
```

```
Best Mean Accuracy: 0.030
Best config {'anova__k': 4}
```

[75]:
```
#comparing different no. of features selected using ANOVA f-test
from numpy import mean
from numpy import std
from sklearn.model_selection import cross_val_score
```

[88]:
```
def evaluate_model(model):
 cv=RepeatedStratifiedKFold(n_repeats=3,n_splits=2, random_state=1)
 scores=cross_val_score(model,X,y,scoring='accuracy',cv=cv,n_jobs=1)
 return scores
```

[77]:
```
#define the data set
X,y=load_dataset('/content/drive/MyDrive/Project/amazon(DA).csv')
```

[96]:
```
#define the no. of features
num_features=[i+1 for i in range(X.shape[1])]
num_features
```

[96]: [1, 2, 3, 4]

[79]:
```
#enumerate each no. of feature
results=list()
for k in num_features:
#create pipeline
 model=LogisticRegression(solver='liblinear')
 fs=SelectKBest(score_func=f_classif,k=k)
 pipeline=Pipeline(steps=[('anova',fs),('lr',model)])
```

[104]:
```
#evaluate the model
scores=evaluate_model(pipeline)
print(results.append(scores))
#results.shape
```

```
/usr/local/lib/python3.9/dist-packages/sklearn/model_selection/_split.py:700:
UserWarning: The least populated class in y has only 1 members, which is less
than n_splits=2.
  warnings.warn(
/usr/local/lib/python3.9/dist-packages/sklearn/svm/_base.py:1244:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  warnings.warn(
/usr/local/lib/python3.9/dist-packages/sklearn/model_selection/_split.py:700:
UserWarning: The least populated class in y has only 1 members, which is less
```

```
than n_splits=2.
  warnings.warn(
/usr/local/lib/python3.9/dist-packages/sklearn/model_selection/_split.py:700:
UserWarning: The least populated class in y has only 1 members, which is less
than n_splits=2.
  warnings.warn(
/usr/local/lib/python3.9/dist-packages/sklearn/svm/_base.py:1244:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  warnings.warn(

None
```

[112]:
```
print('#%d %.3f (%.3f)' %(k,mean(scores),std(scores)))
pyplot.boxplot(results, labels=num_features, showmeans=True)
pyplot.show()
```

```
#4 0.030 (0.002)
```