feature-selection-6

April 10, 2023

[4]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OrdinalEncoder
from sklearn.preprocessing import LabelEncoder
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
from sklearn.feature_selection import mutual_info_classif
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
```

[2]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call
drive.mount("/content/drive", force_remount=True).
```

[5]:
```
df = pd.read_csv('/content/drive/MyDrive/Data Analytics/person.csv')
df.dropna(inplace = True)
df = df.iloc[:,[0,2,3]]
```

[6]:
```
dataset = df.values
dataset
```

[6]:
```
array([['New York', 'Male', 'Single'],
       ['Toronto', 'Female', 'Married'],
       ['Paris', 'Male', 'Single'],
       ['London', 'Male', 'Single'],
       ['Los Angeles ', 'Female', 'Divorced'],
       ['Tokyo', 'Male', 'Married'],
       ['London', 'Male', 'Single'],
       ['Paris', 'Female', 'Single'],
       ['Chicago', 'Male', 'Married'],
       ['London', 'Male', 'Married'],
       ['Vancouver', 'Female', 'Divorced'],
       ['Paris', 'Female', 'Married'],
       ['Munich', 'Male', 'Single'],
```

```
       ['Tokyo', 'Female', 'Single'],
       ['New York', 'Female', 'Married'],
       ['London', 'Male', 'Married'],
       ['Munich', 'Male', 'Married'],
       ['Tokyo', 'Female', 'Single']], dtype=object)
```

[7]:
```
# Splitting data intp input and output variables
X = dataset[:,:-1]
Y = dataset[:,-1]
```

[8]:
```
# Formatting fields as strings
X = X.astype(str)
```

[9]:
```
# Splitting data in training and testing set
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 (12, 2) (12,)
Test (6, 2) (6,)
```

[10]:
```
# Preparing input variable
def prepare_inputs(X_train, X_test):
  oe = OrdinalEncoder()
  # Fitting encoding on training set
  oe.fit(X_train)
  # Applying on train set
  X_train_enc = oe.transform(X_train)
  # Applying on test set
  X_test_enc=oe.transform(X_test)
  return X_train_enc, X_test_enc
```

[11]:
```
# Preparing target variable
def prepare_target(Y_train, Y_test):
  le = LabelEncoder()
  le.fit(Y_train)
  Y_train_enc = le.transform(Y_train)
  Y_test_enc = le.transform(Y_test)
  return Y_train_enc, Y_test_enc
```

[12]:
```
X_train_enc, X_test_enc = prepare_inputs(X_train, X_test)
Y_train_enc, Y_test_enc = prepare_target(Y_train, Y_test)
print('Train', X_train_enc.shape, Y_train_enc.shape)
print('Test', X_test_enc.shape, Y_test_enc.shape)
```

```
Train (12, 2) (12,)
```

```
Test (6, 2) (6,)
```

1 Chi- Square feature selection

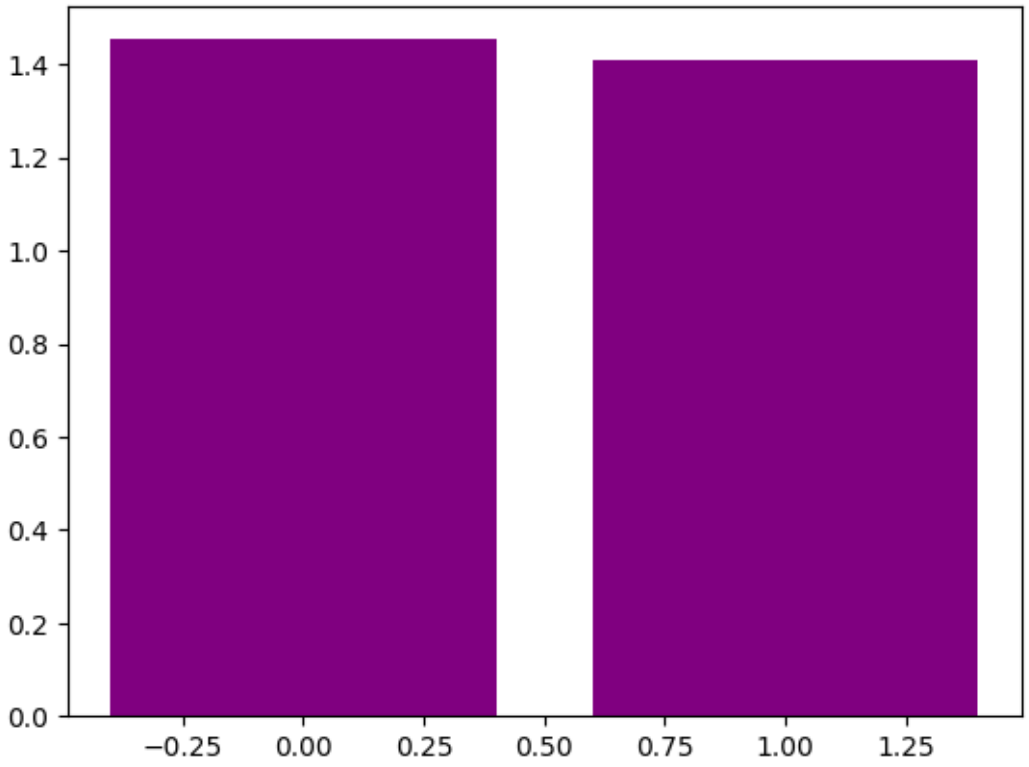
[13]:
```
def select_features(X_train, Y_train, X_test):
  fs = SelectKBest(score_func=chi2, k='all')
  fs.fit(X_train, Y_train)
  X_train_fs = fs.transform(X_train)
  X_test_fs = fs.transform(X_test)
  return X_train_fs, X_test_fs, fs
```

[14]:
```
X_train_fs, X_test_fs, fs = select_features(X_train_enc, Y_train_enc,␣
↪X_test_enc)
```

[15]:
```
fs.scores_
```

[15]:
```
array([1.45341615, 1.40816327])
```

[16]:
```
plt.bar([i for i in range (len(fs.scores_))],fs.scores_, color = 'purple')
plt.show()
```



2 Mutual Information feature selection

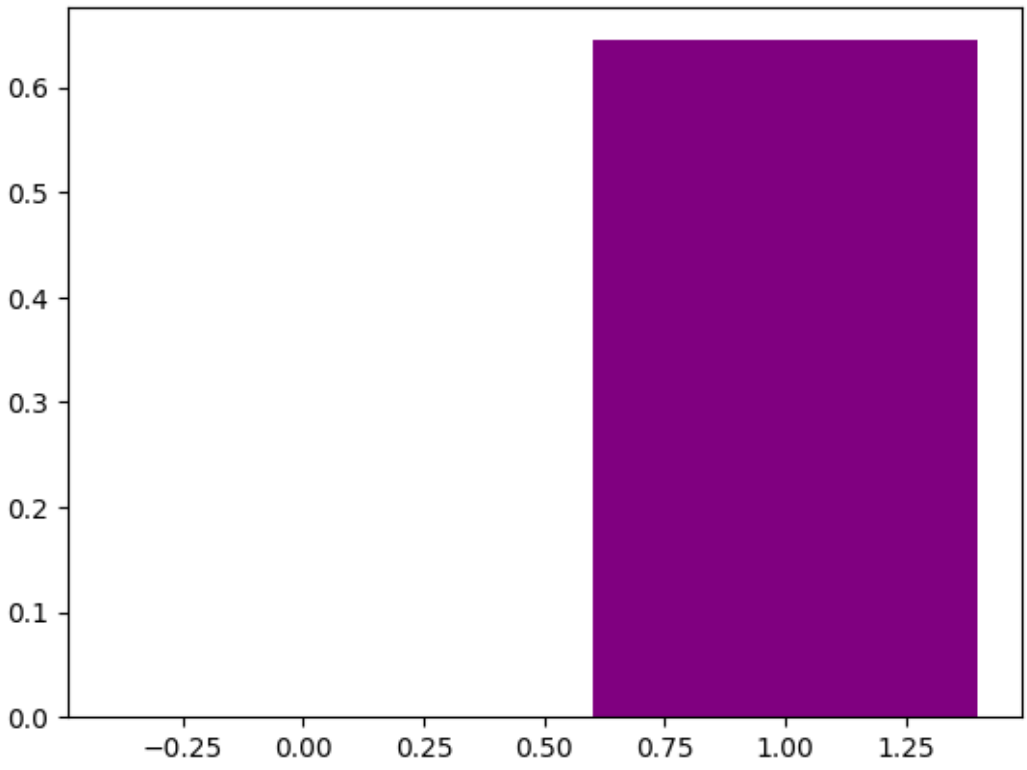
[17]:
```
def select_features_2(X_train, Y_train, X_test):
  fs=SelectKBest(score_func=mutual_info_classif, k='all')
  fs.fit(X_train, Y_train)
  X_train_fs_2=fs.transform(X_train)
  X_test_fs_2=fs.transform(X_test)
  return X_train_fs_2, X_test_fs_2, fs

#Calling feature selection function
X_train_fs_2, X_test_fs_2, fs = select_features_2(X_train_enc, Y_train_enc,␣
 ↪X_test_enc)

fs.scores_
```

[17]:
```
array([0.        , 0.64448052])
```

[18]:
```
plt.bar([i for i in range (len(fs.scores_))],fs.scores_, color = 'purple')
plt.show()
```



3 Model built using all features

[19]:
```
model = LogisticRegression(solver='lbfgs')
model.fit(X_train_enc, Y_train_enc)

LogisticRegression(C=1.0, class_weight = None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio = None, max_iter=100,
multi_class = 'auto', n_jobs=None, penalty = 'l2',
random_state=None, solver='lbfgs', tol=0.0001, verbose=0, warm_start=False)

#predict the model
yhat = model.predict(X_test_enc)
#Evaluate the prediction
accuracy = accuracy_score(Y_test_enc, yhat)
print("Accuracy: %.2f" %(accuracy*100))
```

```
Accuracy: 16.67
```

4 Model built using Chi-squared features

[20]:
```
model1 = LogisticRegression(solver='lbfgs')
#fit the model
model1.fit(X_train_fs, Y_train_enc)
#evaluate the model
yhat = model1.predict(X_test_fs)
#evaluate the performance
accuracy = accuracy_score(Y_test_enc, yhat)
print("Accuracy: %.2f" %(accuracy*100))
```

```
Accuracy: 16.67
```

5 Model built using Mutual Information

[21]:
```
model2 = LogisticRegression(solver = 'lbfgs')
#fit the model
model2.fit(X_train_fs_2, Y_train_enc)
#evaluate the model
yhat = model2.predict(X_test_fs_2)
#evaluate the performance
accuracy = accuracy_score(Y_test_enc, yhat)
print("Accuracy: %.2f" %(accuracy*100))
```

```
Accuracy: 16.67
```

Accuracy of chi-square and mutual information is same