CSE 4020 - MACHINE LEARNING

Lab 29+30

Feature Selection

Submitted by: Alokam Nikhitha(19BCE2555)

Question:

Select the best k features based on statistical tests.

- Display the feature importance score (numeric and graph).
- Select the best five features and display their name.

  For regression: f_regression, mutual_info_regression

  For classification: chi2, mutual_info_classif

Dataset Used:

https://www.kaggle.com/uciml/autompg-dataset

Procedure:

- We first import the dataset into our workspace the use of pandas.
- Then we fill missing values using Mean Strategy
- We divide dependent and independent variables
- We split the data into training and Testing sets
- **Then we use f_regression to identify the most co-related five attributes in our dataset**
- **Next, we use mutual_info_regression to identify the most co-related five attributes in our dataset.**
- And we find scores of each attribute
- And make plots

Code Snippets and Explanation:

In [1]:
```
#importing Libraries
import numpy as np
import pandas as pd
```

Here we are importing the libraries.

In [2]:
```
#impoprting Dataset
dataset = pd.read_csv("auto-mpg.csv")
```

We're importing the dataset into our workspace

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

Out[3]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin	car name
0	18.0	8	307.0	130	3504	12.0	70	1	chevrolet chevelle malibu
1	15.0	8	350.0	165	3693	11.5	70	1	buick skylark 320
2	18.0	8	318.0	150	3436	11.0	70	1	plymouth satellite
3	16.0	8	304.0	150	3433	12.0	70	1	amc rebel sst
4	17.0	8	302.0	140	3449	10.5	70	1	ford torino

Printing the first few rows of the Dataset

In [4]:
```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 398 entries, 0 to 397
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   mpg           398 non-null    float64
 1   cylinders     398 non-null    int64  
 2   displacement  398 non-null    float64
 3   horsepower    398 non-null    object 
 4   weight        398 non-null    int64  
 5   acceleration  398 non-null    float64
 6   model year    398 non-null    int64  
 7   origin        398 non-null    int64  
 8   car name      398 non-null    object 
dtypes: float64(3), int64(4), object(2)
memory usage: 28.1+ KB
```

Info of the dataset

In [5]:
```
s=0
for i in dataset['horsepower']:
    if (i!='?'):
        s+=int(i);
print(s)
```

```
40952
```

In [6]:
```
#data shape
dataset['horsepower'].shape
```

Out[6]: (398,)

In [7]:
```
dataset['horsepower'][0]
```

Out[7]: '130'

In [8]:
```
count=0
for i in dataset['horsepower']:
    if(i=='?'):
        dataset['horsepower'][count] = s/398
    count=count+1
```

Here **we're first off** filling **withinside the missing** values **withinside the** horsepower attribute. We **recognize** that the **missing** values are marked as ? and **we're using mean** as our **replacement** value. Here **we've** calculate the sum of **all of the** values in horsepower barring the ? **after which filled** in **the ones** values divided **through total occurrence** in our ? marked value.

```
In [9]: dataset['horsepower'] = pd.to_numeric(dataset['horsepower'])
```

```
In [10]: dataset.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 398 entries, 0 to 397
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   mpg           398 non-null    float64
 1   cylinders     398 non-null    int64
 2   displacement  398 non-null    float64
 3   horsepower    398 non-null    float64
 4   weight        398 non-null    int64
 5   acceleration  398 non-null    float64
 6   model year    398 non-null    int64
 7   origin        398 non-null    int64
 8   car name      398 non-null    object
dtypes: float64(4), int64(4), object(1)
memory usage: 28.1+ KB
```

Since all the values are filled in, we can now convert it into float datatype and on seeing if the conversion has occurred we can see that the Data type of horse power attribute is not float64.

```
In [11]: X = dataset.iloc[:, 1:8].values
         y = dataset.iloc[:, 0].values
```

```
In [12]: from sklearn.model_selection import train_test_split
         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
```

Here we have defined set of dependent and independent attributes and split into Test and Training data sets.

In [13]:
```python
from sklearn.feature_selection import SelectKBest, f_regression
X_new = SelectKBest(f_regression, k=4).fit_transform(X_train, y_train)
```

In [14]:
```python
X_new[0:5]
```

Out[14]:
```
array([[8.000e+00, 3.180e+02, 1.500e+02, 4.135e+03],
       [4.000e+00, 9.700e+01, 6.000e+01, 1.834e+03],
       [4.000e+00, 9.700e+01, 7.800e+01, 2.188e+03],
       [4.000e+00, 9.700e+01, 4.600e+01, 1.950e+03],
       [4.000e+00, 1.200e+02, 7.400e+01, 2.635e+03]])
```

Here we have used the f_regression method to identify the best 5 columns in our training set on basis of their correlation to y attribute of training set.

The attributes identified are ‘cylinders’, ‘displacement’, ‘horsepower’, ‘weight’ and ‘origin’.

In [15]:
```python
from scipy.stats.stats import pearsonr
out_list = []
for i in range(0,7):
    corr_tuple = pearsonr(X_train[:, i], y_train)
    out_list.append([i, corr_tuple[0], corr_tuple[1]])
```

In [16]:
```python
out_list
```

Out[16]:
```
[[0, -0.7661443790538411, 1.2313490354826512e-62],
 [1, -0.7989521996415161, 9.18027655330086e-72],
 [2, -0.7869828295464115, 3.016344048355765e-68],
 [3, -0.8237643945670992, 7.098435723687908e-80],
 [4, 0.4628071406663036, 2.7644724966194742e-18],
 [5, 0.5732556997398187, 3.5600029155622773e-29],
 [6, 0.5742490165089554, 2.717164301894552e-29]]
```

Similarly, we have used mutual_info_regression. The identified best attributes are ‘cylinders’, ‘displacement’, ‘horsepower’, ‘weight’ and ‘model year’.

```
In [17]: corr_dataframe = pd.DataFrame(out_list, columns=["Column Index", "Correlation", "P-value"])
```

```
In [18]: corr_dataframe
```

Out[18]:

	Column Index	Correlation	P-value
0	0	-0.766144	1.231349e-62
1	1	-0.798952	9.180277e-72
2	2	-0.786983	3.016344e-68
3	3	-0.823764	7.098436e-80
4	4	0.462807	2.764472e-18
5	5	0.573256	3.560003e-29
6	6	0.574249	2.717164e-29

Here **we're locating** the Pearson's coefficient and p **fee of every** attribute. The **decrease** the p **fee** the **higher** the correlation and **better** the **importance** of that attribute.

```
In [19]: corr_dataframe.sort_values(by=['P-value'], inplace=True)
```

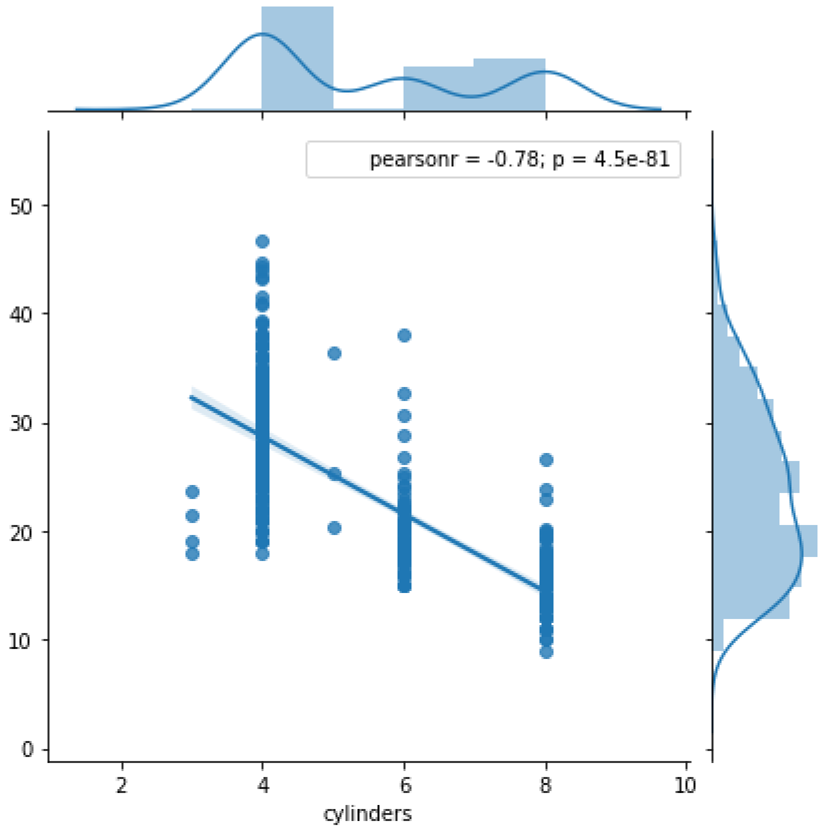
```
In [20]: corr_dataframe
```

Out[20]:

	Column Index	Correlation	P-value
3	3	-0.823764	7.098436e-80
1	1	-0.798952	9.180277e-72
2	2	-0.786983	3.016344e-68
0	0	-0.766144	1.231349e-62
6	6	0.574249	2.717164e-29
5	5	0.573256	3.560003e-29
4	4	0.462807	2.764472e-18

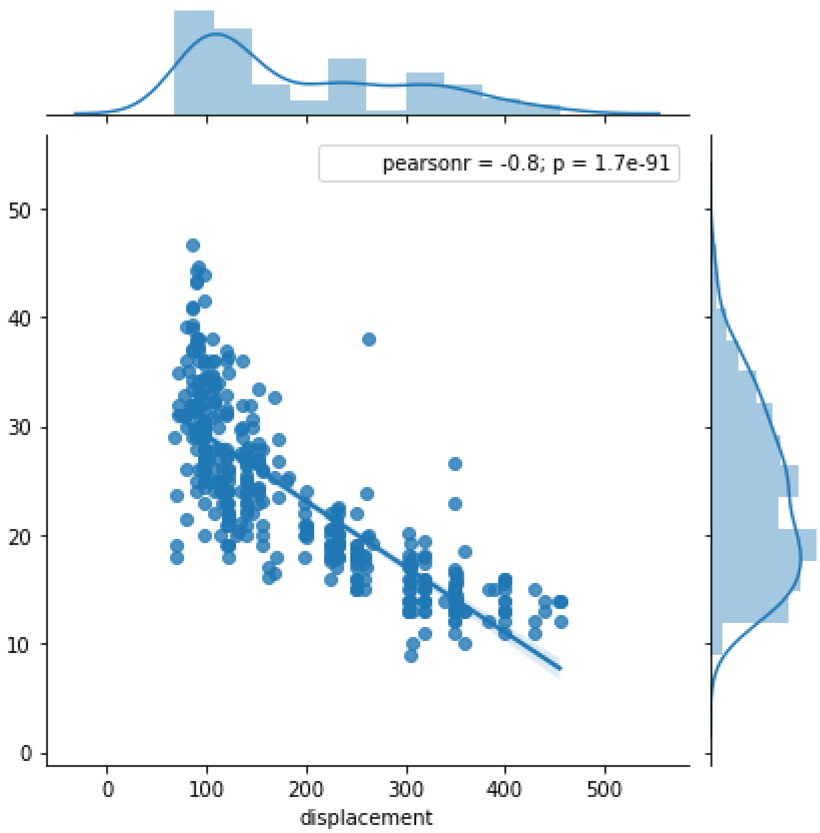
In [21]:
```
import seaborn as sns
j = sns.jointplot("cylinders", y, data=dataset, kind='reg')
j.annotate(pearsonr)
```

Out[21]: <seaborn.axisgrid.JointGrid at 0x1df3a959c70>



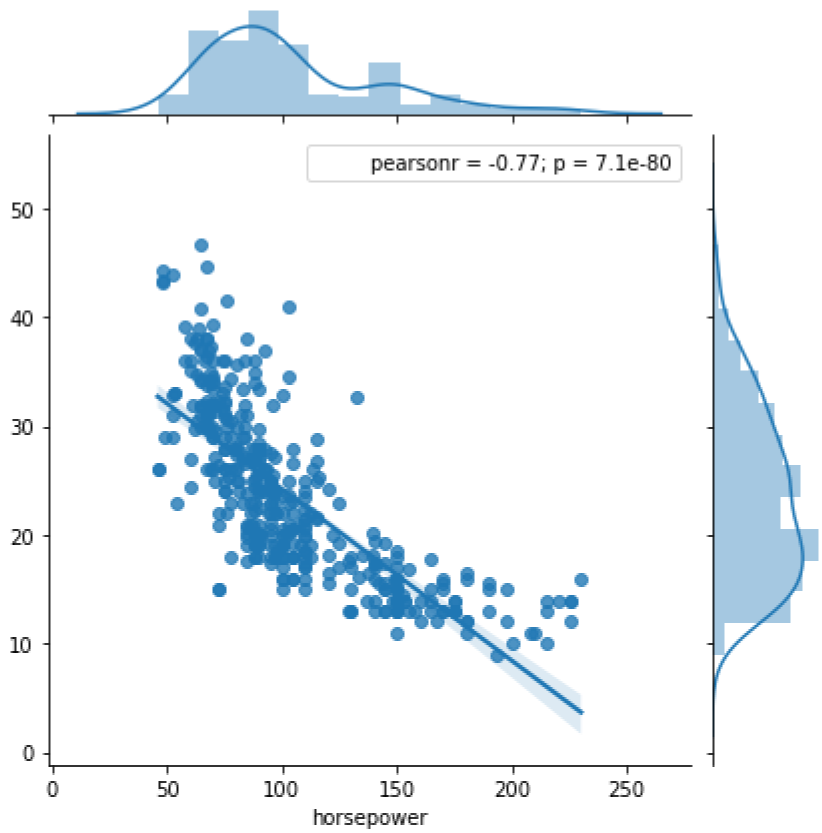
In [22]:
```
j = sns.jointplot("displacement", y, data=dataset, kind='reg')
j.annotate(pearsonr)
```

Out[22]: <seaborn.axisgrid.JointGrid at 0x1df3e8c80a0>



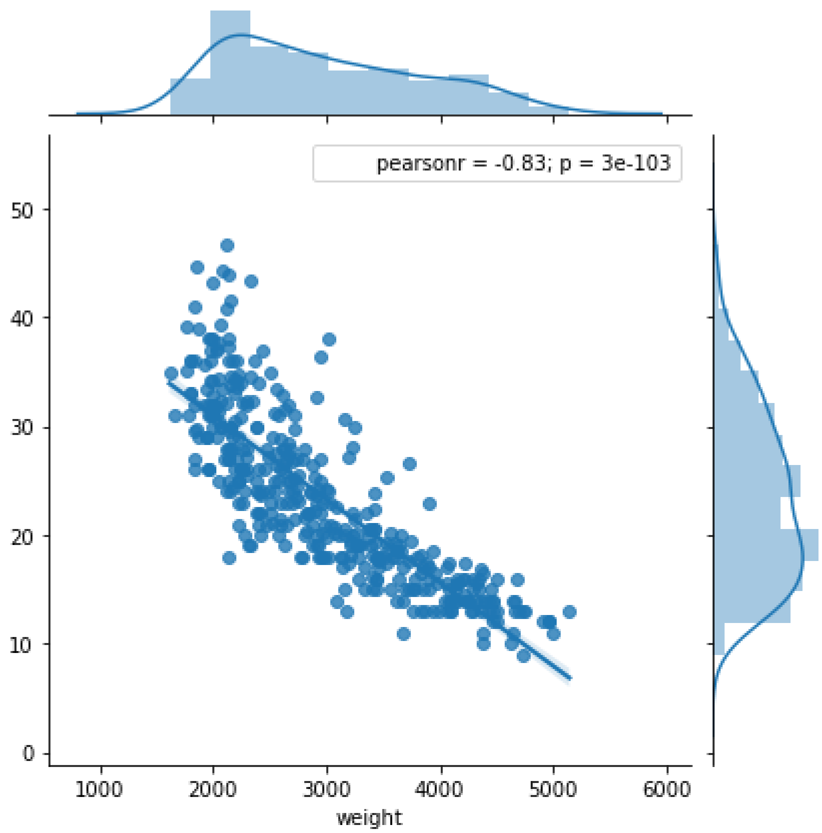
In [23]:
```
j = sns.jointplot("horsepower", y, data=dataset, kind='reg')
j.annotate(pearsonr)
```

Out[23]: <seaborn.axisgrid.JointGrid at 0x1df1b45a220>



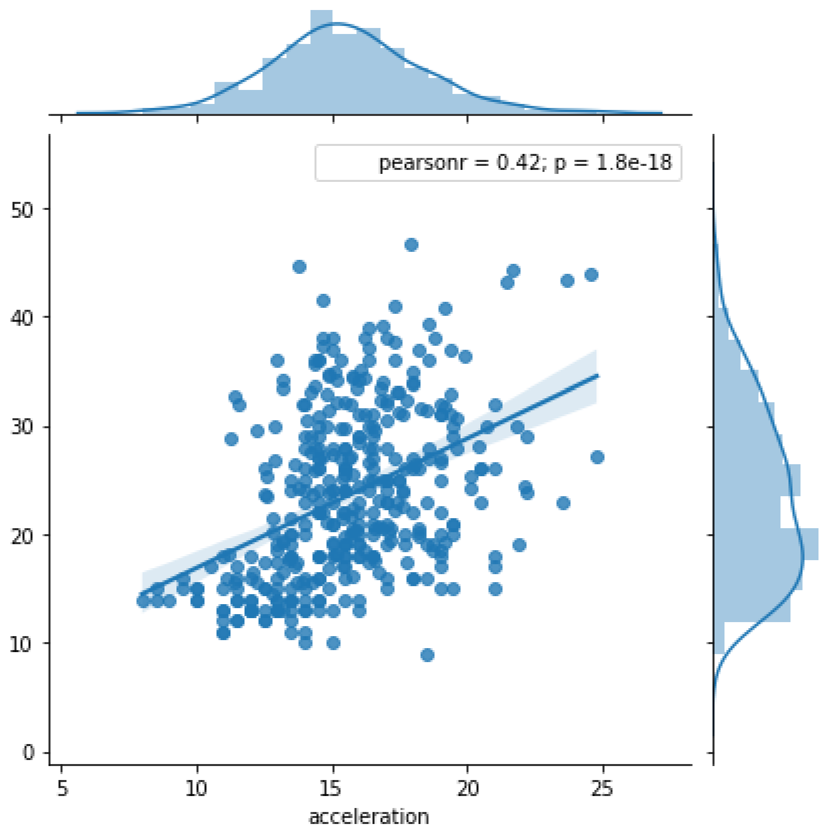
In [24]:
```
j = sns.jointplot("weight", y, data=dataset, kind='reg')
j.annotate(pearsonr)
```

Out[24]: <seaborn.axisgrid.JointGrid at 0x1df3eb05f10>



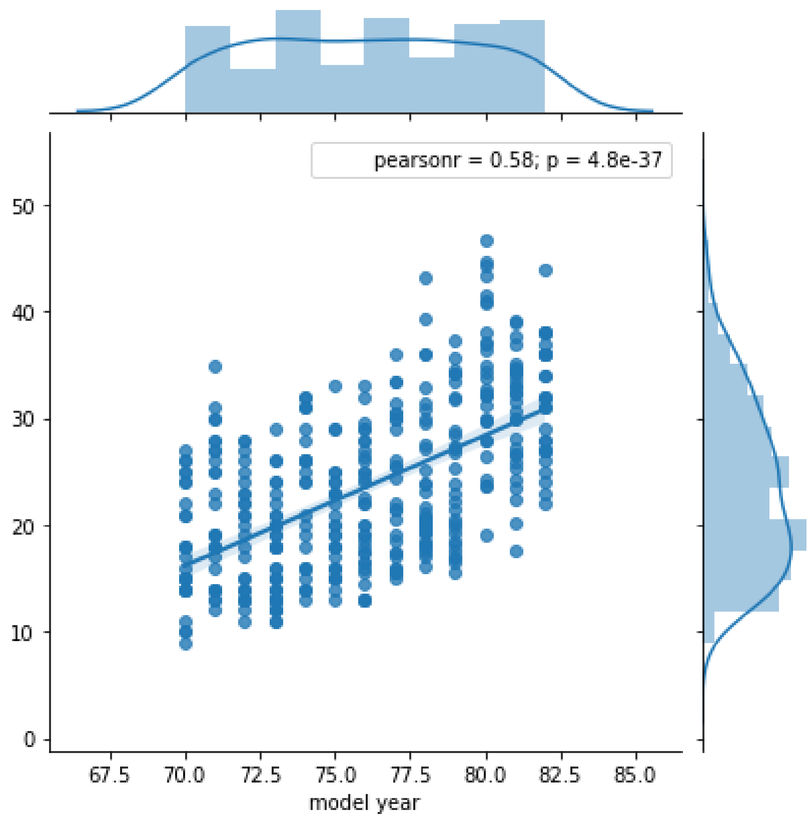
```
In [25]: j = sns.jointplot("acceleration", y, data=dataset, kind='reg')
         j.annotate(pearsonr)
```

```
Out[25]: <seaborn.axisgrid.JointGrid at 0x1df3ec108b0>
```



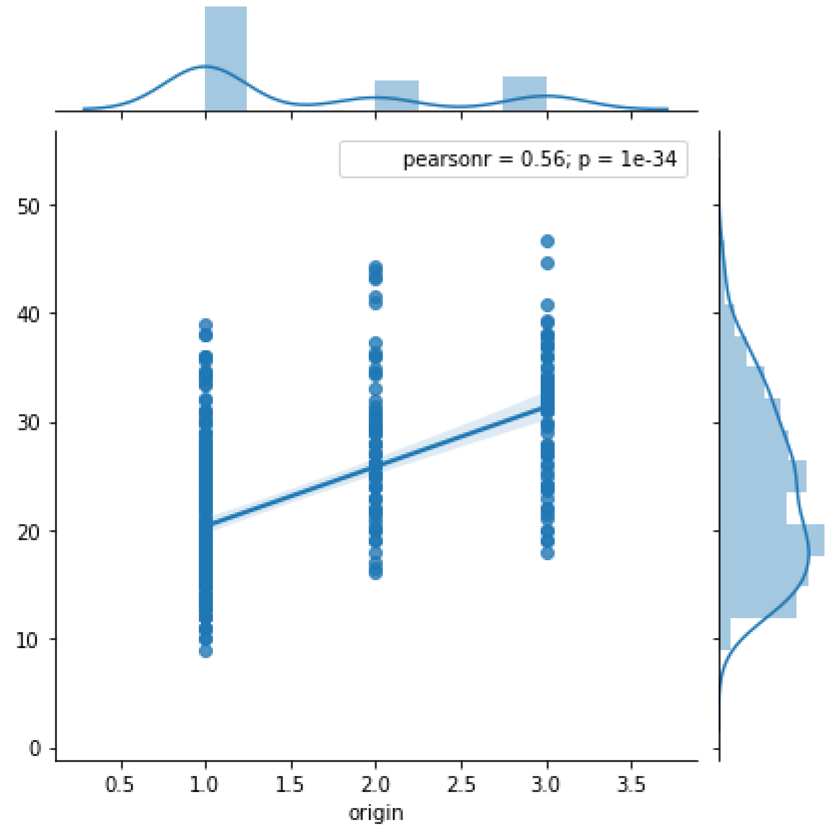
In [26]:
```
j = sns.jointplot("model year", y, data=dataset, kind='reg')
j.annotate(pearsonr)
```

Out[26]: <seaborn.axisgrid.JointGrid at 0x1df3ed3c520>



In [27]:
```
j = sns.jointplot("origin", y, data=dataset, kind='reg')
j.annotate(pearsonr)
```

Out[27]: `<seaborn.axisgrid.JointGrid at 0x1df3edd4bb0>`

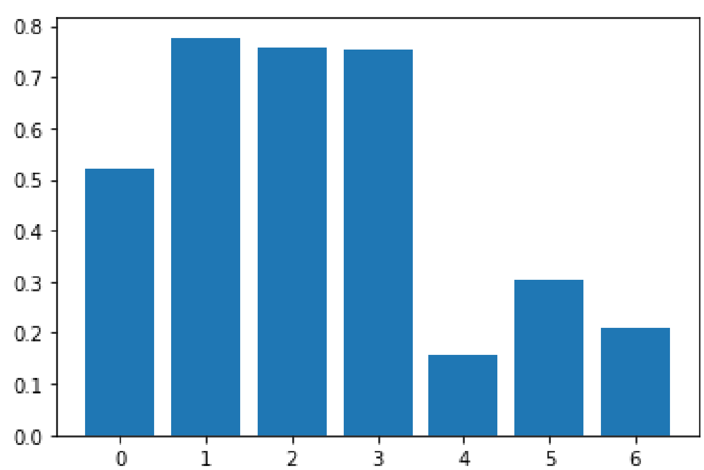


Here we have visualised all the attributes and their correlation with increase/decrease in y values of training set. We can see that attribute including ‘cylinders’, ‘displacement’, ‘horsepower’ and ‘weight’ are highly correlated with y values. There is not much correlation between the other attributes but the closes to fifth spot comes in with ‘origin’ and ‘model year’ attributes.

In [30]:
```python
def select_features(X_train, y_train, X_test):
    fs = SelectKBest(score_func=mutual_info_regression, 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
```

In [31]:
```python
X_train_fs, X_test_fs, fs = select_features(X_train, y_train, X_test)
for i in range(len(fs.scores_)):
    print('Feature %d: %f' % (i, fs.scores_[i]))
plt.bar([i for i in range(len(fs.scores_))], fs.scores_)
plt.show()
```

```
Feature 0: 0.520581
Feature 1: 0.776781
Feature 2: 0.755587
Feature 3: 0.752368
Feature 4: 0.155000
Feature 5: 0.303864
Feature 6: 0.209670
```



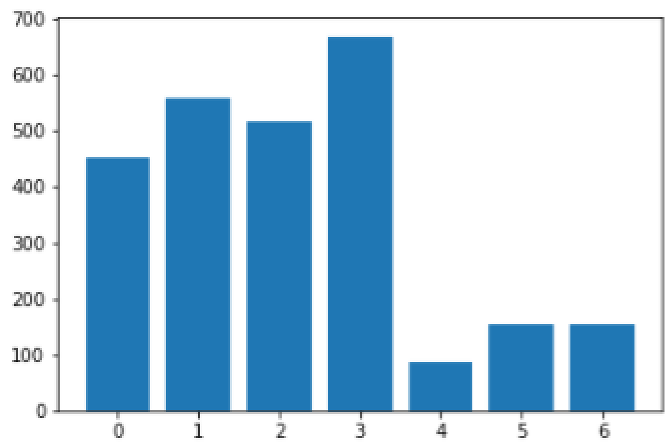
Here we are trying to plot the scores graph of mutual_info_regression.

We can see that the attributes with highest scores in order are:

1. **Displacement**
2. **Horsepower**
3. **Weight**
4. **Cylinders and**
5. **Model Year**

In [32]:
```
fs = SelectKBest(score_func=f_regression, k='all')
fs.fit(X_train, y_train)
X_train_fs = fs.transform(X_train)
X_test_fs = fs.transform(X_test)
for i in range(len(fs.scores_)):
    print('Feature %d: %f' % (i, fs.scores_[i]))
plt.bar([i for i in range(len(fs.scores_))], fs.scores_)
plt.show()
```

```
Feature 0: 449.090952
Feature 1: 557.711663
Feature 2: 514.141435
Feature 3: 667.161119
Feature 4: 86.133061
Feature 5: 154.673817
Feature 6: 155.474265
```



Similarly, here we have plotted the numerical and graphical values of scores. The highest scores in order here are:

1. **Weight**
2. **Displacement**
3. **Horsepower**
4. **Cylinders and**
5. **Origin**

```
In [33]: X_train
```

```
Out[33]: array([[  8. , 318. , 150. , ...,  13.5,  72. ,   1. ],
                [  4. ,  97. ,  60. , ...,  19. ,  71. ,   2. ],
                [  4. ,  97. ,  78. , ...,  15.8,  80. ,   2. ],
                ...,
                [  4. ,  68. ,  49. , ...,  19.5,  73. ,   2. ],
                [  6. , 250. , 100. , ...,  15. ,  71. ,   1. ],
                [  4. ,  90. ,  71. , ...,  16.5,  75. ,   2. ]])
```

Results and Conclusion

The five features according to f_regression are:

```
Feature 0: 449.090952
Feature 1: 557.711663
Feature 2: 514.141435
Feature 3: 667.161119
Feature 4: 86.133061
Feature 5: 154.673817
Feature 6: 155.474265
```

- **Weight → 667.16**
- **Displacement→ 557.71**
- **Horsepower→ 514.14**
- **Cylinders → 449.09**
- **Origin→ 155.47**

The five features according to mutual_info_regression are:

```
Feature 0: 0.520581
Feature 1: 0.776781
Feature 2: 0.755587
Feature 3: 0.752368
Feature 4: 0.155000
Feature 5: 0.303864
Feature 6: 0.209670
```

- **Displacement→ 0.7767**
- **Horsepower→ 0.7558**
- **Weight→ 0.7523**
- **Cylinders → 0.5205**
- **Model Year→ 0.3038**