Feature Selection Techniques 5/5

Feature Selection using Chi - Square Test

Practical Implementation
By Sahil Josan

Compute chi-squared stats between each non-negative feature and class.

- This score should be used to evaluate categorical variables in a classification task.

This score can be used to select the n_features features with the highest values for the test chi-squared statistic from X, which must contain only non-negative features such as booleans or frequencies (e.g., term counts in document classification), relative to the classes.

Recall that the chi-square test measures dependence between stochastic variables, so using this function "weeds out" the features that are the most likely to be independent of class and therefore irrelevant for classification. The Chi Square statistic is commonly used for testing relationships between categorical variables.

It compares the observed distribution of the different classes of target Y among the different categories of the feature, against the expected distribution of the target classes, regardless of the feature categories.

Youtube Videos

Statistical test: Link

https://www.youtube.com/watch?v=YrhlQB3mQFI

In [ ]:
```
import seaborn as sns
df = sns.load_dataset('titanic')
print(f"The dataset has {df.shape[0]} rows and {df.shape[1]} columns ")
df.head()
```

```
The dataset has 891 rows and 15 columns
```

Out[ ]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_t
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	Southam
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherb
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	Southam
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	Southam
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	Southam

In [ ]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 15 columns):
 #   Column       Non-Null Count  Dtype   
---  ------       --------------  -----   
 0   survived     891 non-null    int64   
 1   pclass       891 non-null    int64   
 2   sex          891 non-null    object  
 3   age          714 non-null    float64 
 4   sibsp        891 non-null    int64   
 5   parch        891 non-null    int64   
 6   fare         891 non-null    float64 
 7   embarked     889 non-null    object  
 8   class        891 non-null    category
 9   who          891 non-null    object  
 10  adult_male   891 non-null    bool    
 11  deck         203 non-null    category
 12  embark_town  889 non-null    object  
 13  alive        891 non-null    object  
 14  alone        891 non-null    bool    
dtypes: bool(2), category(2), float64(2), int64(4), object(5)
memory usage: 80.7+ KB
```

In [ ]:
```
df.columns
```

Out[ ]:
```
Index(['survived', 'pclass', 'sex', 'age', 'sibsp', 'parch', 'fare',
       'embarked', 'class', 'who', 'adult_male', 'deck', 'embark_town',
       'alive', 'alone'],
      dtype='object')
```

In [ ]:
```
# Categorical features that we use ["sex","embarked","alone","pclass","survived"]
df = df[["sex","embarked","alone","pclass","survived"]]
df.head()
```

Out[ ]:

	sex	embarked	alone	pclass	survived
0	male	S	False	3	0
1	female	C	False	1	1
2	female	S	True	3	1
3	female	S	False	1	1
4	male	S	True	3	0

Before applying chi-square test, we have to apply label encoding on every feature

In [ ]:
```
# Label Encoding on "sex" column
import numpy as np
df['sex'] = np.where(df['sex'] == 'male',1,0)
```

```
# Label encoding on "embarked" column
ordinal_label = {k:i for i,k in enumerate(df['embarked'].unique(),0)}
```

In [ ]:
```
ordinal_label
```

Out[ ]:
```
{'S': 0, 'C': 1, 'Q': 2, nan: 3}
```

We want to replace

0 where value is S

1 where value is C

2 where value is Q

3 where value in nan

In [ ]:
```
df['embarked'] = df['embarked'].map(ordinal_label)
```

In [ ]:
```
# Label encoding on "alone" column
df['alone'] = np.where(df['alone']==False,0,1)
```

In [ ]:
```
df.head()
```

Out[ ]:

	sex	embarked	alone	pclass	survived
0	1	0	0	3	0
1	0	1	0	1	1
2	0	0	1	3	1
3	0	0	0	1	1
4	1	0	1	3	0

In [ ]:
```
df['sex'].unique()
```

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

Train Test Split

In [ ]:
```
### Train test split is usually done to avoid overfitting
X = df.drop('survived',axis = 1)
y = df['survived']

from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X , y, test_size=0.30, random_state=0)
```

In [ ]:
```
X_train['sex'].unique()
```

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

In [ ]:
```
X_train.shape,y_train.shape
```

Out[ ]:
```
((623, 4), (623,))
```

Perform Chi-Square test

In [ ]:
```
from sklearn.feature_selection import chi2

f_p_values = chi2(X_train,y_train)
f_p_values
```

Out[ ]:
```
(array([63.55447864, 11.83961845,  9.03328564, 21.61080949]),
 array([1.55992554e-15, 5.79837058e-04, 2.65107556e-03, 3.33964360e-06]))
```

- Chi2 returns 2 values `Fscore` and `Pvalue`
- The 1st value in the array is Fscore
- and 2nd value in the array is Pvalue

Note

- The more higher is the value of F-score, more important is the feature is
- Similarly the lesser the P-value, the more important te feature is

Now to practically check, which feature is more important

In [ ]:
```
import pandas as pd
p_values = pd.Series(f_p_values[1])  # P-value
p_values.index = X_train.columns
p_values.sort_values(ascending = False)
```

Out[ ]:
```
alone       2.651076e-03
embarked    5.798371e-04
pclass      3.339644e-06
sex         1.559926e-15
dtype: float64
```

Observation

"Sex" Column is the most important columns when compared to the output feature "survived"