IBM Innovation Centre for Education

Feature Selection- Information Gain

Estimated time needed: **20** minutes

Objectives

After completing this lab you will have a good understanding in:

- Entropy Test
- Mututal Information are not important

This Notebook is created for Python Module

Mutual Information

Mutual information (MI) between two random variables is a non-negative value, which measures the dependency between the variables. It is equal to zero if and only if two random variables are independent, and higher values mean higher dependency. The function relies on nonparametric methods based on entropy estimation from k-nearest neighbors distances.

A quantity called **mutual information** measures the amount of information one can obtain from one random variable given another. The mutual information between two random variables X and Y can be stated formally as follows:

I(X ; Y) = H(X) – H(X | Y)

Where I(X ; Y) is the mutual information for X and Y, H(X) is the entropy for X and H(X | Y) is the conditional entropy for X given Y. The result has the units of bits.

You can use information gain to decide which attribute goes at which level in dataset. By using information gain as a criterion, we try to estimate the information contained by each attribute. To measure the randomness or uncertainty of a random variable X is called Entropy. By calculating the entropy measure of each attribute we can calculate their information gain. Information Gain calculates the expected reduction in entropy due to sorting on the attribute.

For a binary classification problem with only two classes, positive and negative class.

- If all examples are positive or all are negative then entropy will be zero i.e, low.
- If half of the records are of positive class and half are of negative class then entropy is one i.e, high.

$H(X) = \mathrm{E}_X[I(x)] = -\sum_{x \in \mathrm{X}} p(x) \log p(x).$

By calculating the entropy measure of each attribute we can calculate their information gain. Information Gain calculates the expected reduction in entropy due to sorting on the attribute. Information gain can be calculated.

Entropy

Entropy is defined as the amount of uncertainty/randomness in the data; the greater the randomness, the greater the entropy. To make decisions, information gain employs entropy. Information increases as entropy decreases. In decision trees and random forests, information gain is used to determine the best split. As a result, the greater the information gain, the better the split, and thus the lower the entropy.

To calculate **information gain**, the entropy of a dataset before and after a split is used.

Information gain for Classification Dataset

- Dataset : Wine

In [11]:

```
import pandas as pd
from sklearn.datasets import load_wine
df = load_wine()
print(df.DESCR)
```

```
.. _wine_dataset:

Wine recognition dataset
------------------------
```

```
**Data Set Characteristics:**

    :Number of Instances: 178 (50 in each of three classes)
    :Number of Attributes: 13 numeric, predictive attributes and the class
    :Attribute Information:
                - Alcohol
                - Malic acid
                - Ash
                - Alcalinity of ash
                - Magnesium
                - Total phenols
                - Flavanoids
                - Nonflavanoid phenols
                - Proanthocyanins
                - Color intensity
                - Hue
                - OD280/OD315 of diluted wines
                - Proline

    - class:
            - class_0
            - class_1
            - class_2

    :Summary Statistics:

    ============================= ==== ===== ======= =====
                                   Min   Max   Mean     SD
    ============================= ==== ===== ======= =====
    Alcohol:                      11.0  14.8    13.0   0.8
    Malic Acid:                   0.74  5.80    2.34  1.12
    Ash:                          1.36  3.23    2.36  0.27
    Alcalinity of Ash:            10.6  30.0    19.5   3.3
    Magnesium:                    70.0 162.0    99.7  14.3
    Total Phenols:                0.98  3.88    2.29  0.63
    Flavanoids:                   0.34  5.08    2.03  1.00
    Nonflavanoid Phenols:         0.13  0.66    0.36  0.12
    Proanthocyanins:              0.41  3.58    1.59  0.57
    Colour Intensity:              1.3  13.0     5.1   2.3
    Hue:                          0.48  1.71    0.96  0.23
    OD280/OD315 of diluted wines: 1.27  4.00    2.61  0.71
    Proline:                       278  1680     746   315
    ============================= ==== ===== ======= =====

    :Missing Attribute Values: None
    :Class Distribution: class_0 (59), class_1 (71), class_2 (48)
    :Creator: R.A. Fisher
    :Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)
    :Date: July, 1988

This is a copy of UCI ML Wine recognition datasets.
https://archive.ics.uci.edu/ml/machine-learning-databases/wine/wine.data

The data is the results of a chemical analysis of wines grown in the same
region in Italy by three different cultivators. There are thirteen different
measurements taken for different constituents found in the three types of
wine.

Original Owners:

Forina, M. et al, PARVUS -
An Extendible Package for Data Exploration, Classification and Correlation.
Institute of Pharmaceutical and Food Analysis and Technologies,
Via Brigata Salerno, 16147 Genoa, Italy.

Citation:

Lichman, M. (2013). UCI Machine Learning Repository
[https://archive.ics.uci.edu/ml]. Irvine, CA: University of California,
School of Information and Computer Science.

.. topic:: References

  (1) S. Aeberhard, D. Coomans and O. de Vel,
  Comparison of Classifiers in High Dimensional Settings,
  Tech. Rep. no. 92-02, (1992), Dept. of Computer Science and Dept. of
  Mathematics and Statistics, James Cook University of North Queensland.
  (Also submitted to Technometrics).

  The data was used with many others for comparing various
  classifiers. The classes are separable, though only RDA
  has achieved 100% correct classification.
  (RDA : 100%, QDA 99.4%, LDA 98.9%, 1NN 96.1% (z-transformed data))
  (All results using the leave-one-out technique)

  (2) S. Aeberhard, D. Coomans and O. de Vel,
  "THE CLASSIFICATION PERFORMANCE OF RDA"
  Tech. Rep. no. 92-01, (1992), Dept. of Computer Science and Dept. of
  Mathematics and Statistics, James Cook University of North Queensland.
  (Also submitted to Journal of Chemometrics).
```

In [12]:

```
wine = pd.DataFrame(df.data, columns=df.feature_names)
```

In [13]:

```
df.target_names
```

Out[13]:

```
array(['class_0', 'class_1', 'class_2'], dtype='<U7')
```

In [18]:

```
wine['Wine']=df.target
wine.head(3)
```

Out[18]:

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid_phenols	proanthocyanins	color_
0	14.23	1.71	2.43	15.6	127.0	2.80	3.06	0.28	2.29	
1	13.20	1.78	2.14	11.2	100.0	2.65	2.76	0.26	1.28	
2	13.16	2.36	2.67	18.6	101.0	2.80	3.24	0.30	2.81	

In [19]:

```
### Train test split to avoid overfitting
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(wine.drop(labels=['Wine'], axis=1),
    wine['Wine'],
    test_size=0.3,
    random_state=0)
```

In [20]:

```
X_train.head()
```

Out[20]:

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid_phenols	proanthocyanins	col
22	13.71	1.86	2.36	16.6	101.0	2.61	2.88	0.27	1.69	
108	12.22	1.29	1.94	19.0	92.0	2.36	2.04	0.39	2.08	
175	13.27	4.28	2.26	20.0	120.0	1.59	0.69	0.43	1.35	
145	13.16	3.57	2.15	21.0	102.0	1.50	0.55	0.43	1.30	
71	13.86	1.51	2.67	25.0	86.0	2.95	2.86	0.21	1.87	

In [21]:

```
from sklearn.feature_selection import mutual_info_classif
# determine the mutual information
mutual_info = mutual_info_classif(X_train, y_train)
mutual_info
```

Out[21]:

```
array([0.41293704, 0.30827622, 0.15896926, 0.28180259, 0.17928789,
       0.49486399, 0.71442787, 0.13178006, 0.26673967, 0.61369166,
       0.54255607, 0.55676134, 0.53368383])
```

In [22]:

```
mutual_info = pd.Series(mutual_info)
mutual_info.index = X_train.columns
mutual_info.sort_values(ascending=False)
```

Out[22]:

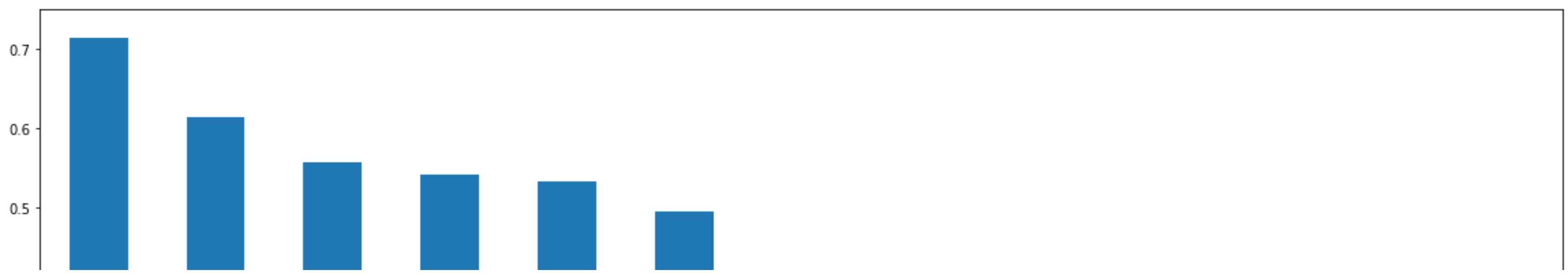
```
flavanoids                      0.714428
color_intensity                 0.613692
od280/od315_of_diluted_wines    0.556761
hue                             0.542556
proline                         0.533684
total_phenols                   0.494864
alcohol                         0.412937
malic_acid                      0.308276
alcalinity_of_ash               0.281803
proanthocyanins                 0.266740
magnesium                       0.179288
ash                             0.158969
nonflavanoid_phenols            0.131780
dtype: float64
```

In [23]:

```
#let's plot the ordered mutual_info values per feature
mutual_info.sort_values(ascending=False).plot.bar(figsize=(20, 8))
```

Out[23]:

```
<AxesSubplot:>
```



[Note]: In Classification we use Mutual info Classif and For Selection we use SelectKBest

In [24]:
```
from sklearn.feature_selection import SelectKBest
```

In [25]:
```
#No we Will select the  top 5 important features
sel_five_cols = SelectKBest(mutual_info_classif, k=5)
sel_five_cols.fit(X_train, y_train)
X_train.columns[sel_five_cols.get_support()]
```

Out[25]:
```
Index(['flavanoids', 'color_intensity', 'hue', 'od280/od315_of_diluted_wines',
       'proline'],
      dtype='object')
```

Now Let's Perform Information Gain on Regression dataset

- Dataset : California Housing

In [27]:
```
from sklearn.datasets import fetch_california_housing
housing = fetch_california_housing()
#from sklearn.datasets import fetch_openml
#housing = fetch_openml(name="house_prices", as_frame=True)
df = pd.DataFrame(housing.data, columns = housing.feature_names)
df["HOUSE_VAL"] = housing.target
```

About the dataset

In [44]:
```python
print(housing.DESCR)
```

```
.. _california_housing_dataset:

California Housing dataset
--------------------------

**Data Set Characteristics:**

    :Number of Instances: 20640

    :Number of Attributes: 8 numeric, predictive attributes and the target

    :Attribute Information:
        - MedInc        median income in block group
        - HouseAge      median house age in block group
        - AveRooms      average number of rooms per household
        - AveBedrms     average number of bedrooms per household
        - Population    block group population
        - AveOccup      average number of household members
        - Latitude      block group latitude
        - Longitude     block group longitude

    :Missing Attribute Values: None

This dataset was obtained from the StatLib repository.
https://www.dcc.fc.up.pt/~ltorgo/Regression/cal_housing.html

The target variable is the median house value for California districts,
expressed in hundreds of thousands of dollars ($100,000).

This dataset was derived from the 1990 U.S. census, using one row per census
block group. A block group is the smallest geographical unit for which the U.S.
Census Bureau publishes sample data (a block group typically has a population
of 600 to 3,000 people).

An household is a group of people residing within a home. Since the average
number of rooms and bedrooms in this dataset are provided per household, these
columns may take surpinsingly large values for block groups with few households
and many empty houses, such as vacation resorts.

It can be downloaded/loaded using the
:func:`sklearn.datasets.fetch_california_housing` function.

.. topic:: References

    - Pace, R. Kelley and Ronald Barry, Sparse Spatial Autoregressions,
      Statistics and Probability Letters, 33 (1997) 291-297
```

In [29]:
```python
df.head()
```

Out[29]:

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude	HOUSE_VAL
0	8.3252	41.0	6.984127	1.023810	322.0	2.555556	37.88	-122.23	4.526
1	8.3014	21.0	6.238137	0.971880	2401.0	2.109842	37.86	-122.22	3.585
2	7.2574	52.0	8.288136	1.073446	496.0	2.802260	37.85	-122.24	3.521
3	5.6431	52.0	5.817352	1.073059	558.0	2.547945	37.85	-122.25	3.413
4	3.8462	52.0	6.281853	1.081081	565.0	2.181467	37.85	-122.25	3.422

In [30]:
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20640 entries, 0 to 20639
Data columns (total 9 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   MedInc      20640 non-null  float64
 1   HouseAge    20640 non-null  float64
 2   AveRooms    20640 non-null  float64
 3   AveBedrms   20640 non-null  float64
 4   Population  20640 non-null  float64
 5   AveOccup    20640 non-null  float64
 6   Latitude    20640 non-null  float64
 7   Longitude   20640 non-null  float64
 8   HOUSE_VAL   20640 non-null  float64
dtypes: float64(9)
memory usage: 1.4 MB
```

In [38]:
```python
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(df.drop(labels=['HOUSE_VAL'], axis=1),
                                               df['HOUSE_VAL'], test_size=0.3,random_state=0)
```

In [40]:
```python
X_train.head()
```

Out[40]:

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude
1989	1.9750	52.0	2.800000	0.700000	193.0	4.825000	36.73	-119.79
256	2.2604	43.0	3.671480	1.184116	836.0	3.018051	37.77	-122.21
7887	6.2990	17.0	6.478022	1.087912	1387.0	3.810440	33.87	-118.04
4581	1.7199	17.0	2.518000	1.196000	3051.0	3.051000	34.06	-118.28

[Note]: In Regression we use Mutual info Regression and For Selection we use SelectPercentile

In [41]:

```
from sklearn.feature_selection import mutual_info_regression
# determine the mutual information
mutual_info = mutual_info_regression(X_train.fillna(0), y_train)
mutual_info
```

Out[41]:

```
array([0.39202229, 0.03506243, 0.10467856, 0.0331393 , 0.02732682,
       0.06255812, 0.36633757, 0.40030775])
```

In [42]:

```
mutual_info = pd.Series(mutual_info)
mutual_info.index = X_train.columns
mutual_info.sort_values(ascending=False)
```

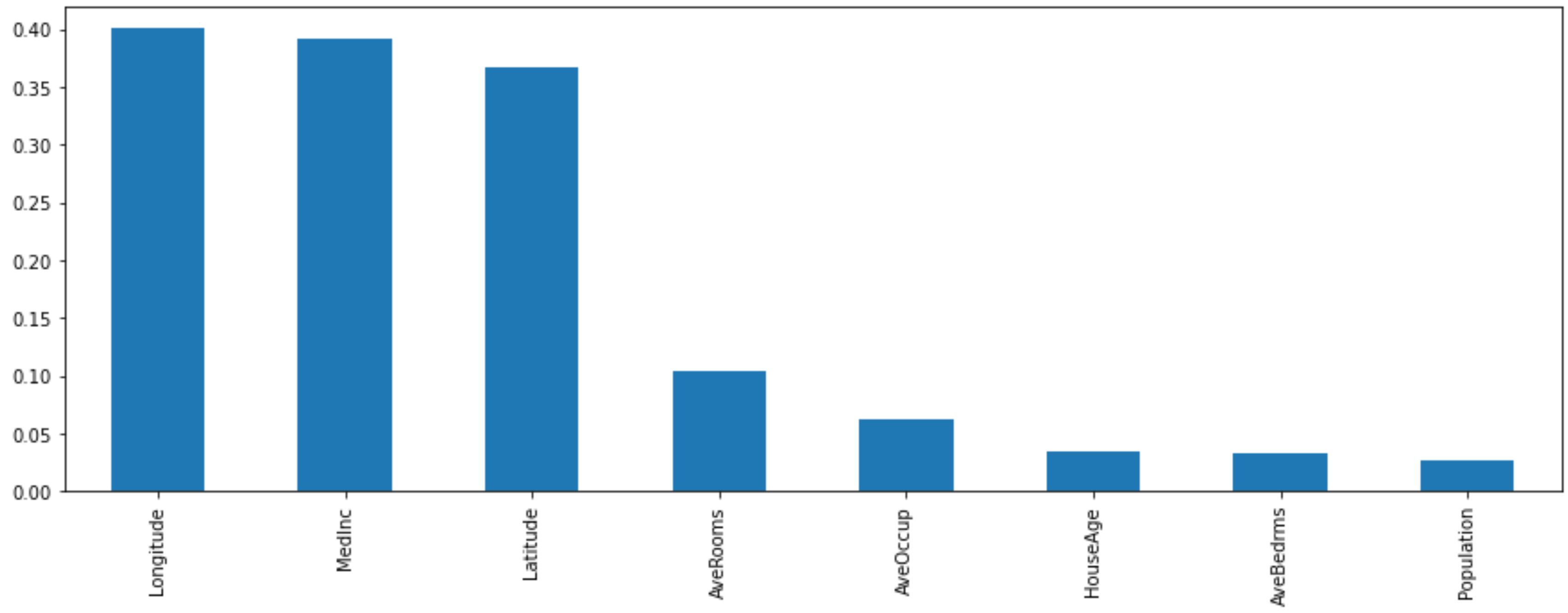
Out[42]:

```
Longitude     0.400308
MedInc        0.392022
Latitude      0.366338
AveRooms      0.104679
AveOccup      0.062558
HouseAge      0.035062
AveBedrms     0.033139
Population    0.027327
dtype: float64
```

In [43]:

```
mutual_info.sort_values(ascending=False).plot.bar(figsize=(15,5))
```

Out[43]: <AxesSubplot:>



In [49]:

```
from sklearn.feature_selection import SelectPercentile
selected_top_columns = SelectPercentile(mutual_info_regression, percentile=20)
selected_top_columns.fit(X_train.fillna(0), y_train)
X_train.columns[selected_top_columns.get_support()]
```

Out[49]:

```
Index(['MedInc', 'Longitude'], dtype='object')
```

Great Job!

That's all we need to know for now! Congratulations, you have learnt one more topic and hands-on with Python.This Notebook is prepared by Sumit Kumar Shukla IBM ICE.

About Author

Mr. Sumit is a Subject Matter Expert at IBM, and a data Scientist with more than five years of experience tutoring students from IITs, NITs, IISc, IIMs, and other prestigious institutions. Google Data Studio certified and IBM certified data analyst Data Science, Machine Learning Models, Graph Databases, and Data Mining techniques for Predictive Modeling and Analytics, as well as data integration, require expertise in Machine Learning and programming languages such as Python, R, and Tableau.

In [ ]: