Experiment

No.8 Title: Attribute subset selection

Batch: A2 **Roll No.: 16010421063** **Experiment No.: 8**

Aim: Attribute subset selection

Resources needed: Python

Theory:

Attribute subset Selection is a technique which is used for data reduction in data mining process. Data reduction reduces the size of data so that it can be used for analysis purposes more efficiently.

Need of Attribute Subset Selection-

The data set may have a large number of attributes. But some of those attributes can be irrelevant or redundant. The goal of attribute subset selection is to find a minimum set of attributes such that dropping of those irrelevant attributes does not much affect the utility of data and the cost of data analysis could be reduced. Mining on a reduced data set also makes the discovered pattern easier to understand.

Methods of Attribute Subset Selection-

1. Stepwise Forward Selection.
2. Stepwise Backward Elimination.
3. Combination of Forward Selection and Backward Elimination.
4. Decision Tree Induction.

All the above methods are greedy approaches for attribute subset selection.

1. **Stepwise Forward Selection:** This procedure start with an empty set of attributes as the minimal set. The most relevant attributes are chosen(having minimum p-value) and are added to the minimal set. In each iteration, one attribute is added to a reduced set.
2. **Stepwise Backward Elimination:** Here all the attributes are considered in the initial set of attributes. In each iteration, one attribute is eliminated from the set of attributes whose p-value is higher than significance level.
3. **Combination of Forward Selection and Backward Elimination:** The stepwise forward selection and backward elimination are combined so as to select the relevant attributes most efficiently. This is the most common technique which is generally used for attribute selection.
4. **Decision Tree Induction:** This approach uses decision tree for attribute selection. It constructs a flow chart like structure having nodes denoting a test on an attribute. Each

branch corresponds to the outcome of test and leaf nodes is a class prediction. The attribute that is not the part of tree is considered irrelevant and hence discarded.

Procedure / Approach /Algorithm / Activity Diagram:

This lab on Subset Selection is a Python adaptation of p. 244-247 of "Introduction to Statistical Learning with Applications in R" by Gareth James, Daniela Witten, Trevor Hastie and Robert Tibshirani. Adapted by R. Jordan Crouser at Smith College for SDS293: Machine Learning (Spring 2016).

Technique used: Best Subset Selection

Url: http://www.science.smith.edu/~jcrouser/SDS293/labs/lab8-py.html

Results: Students must submit the output of above activity.

Files

sample_data

Hitters.csv

```
%matplotlib inline
import pandas as pd
import numpy as np
import itertools
import time
import statsmodels.api as sm
import matplotlib.pyplot as plt
```

```
hitters_df = pd.read_csv('Hitters.csv')
hitters_df.head()
```

HmRun	Runs	RBI	Walks	Years	CAtBat	CHits	CHmRun	CRuns	CRBI	CWalks	League	Division	PutOuts	Assists	Errors	Salary	NewLeague
1	30	29	14	1	293	66	1	30	29	14	A	E	446	33	20	NaN	A
7	24	38	39	14	3449	835	69	321	414	375	N	W	632	43	10	475.0	N
18	66	72	76	3	1624	457	63	224	266	263	A	W	880	82	14	480.0	A
20	65	78	37	11	5628	1575	225	828	838	354	N	E	200	11	3	500.0	N
10	39	42	30	2	396	101	12	48	46	33	N	E	805	40	4	91.5	N

[3]
```
print("Number of null values:", hitters_df["Salary"].isnull().sum())
```

```
Number of null values: 59
```

[5]
```
# Print the dimensions of the original Hitters data (322 rows x 20 columns)
print("Dimensions of original data:", hitters_df.shape)

# Drop any rows the contain missing values, along with the player names
hitters_df_clean = hitters_df.dropna().drop('Walks', axis=1)

# Print the dimensions of the modified Hitters data (263 rows x 20 columns)
print("Dimensions of modified data:", hitters_df_clean.shape)

# One last check: should return 0
print("Number of null values:", hitters_df_clean["Salary"].isnull().sum())
```

```
Dimensions of original data: (322, 20)
Dimensions of modified data: (263, 19)
Number of null values: 0
```

[7]
```
dummies = pd.get_dummies(hitters_df_clean[['League', 'Division', 'NewLeague']])

y = hitters_df_clean.Salary

# Drop the column with the independent variable (Salary), and columns for which we created dummy variables
X_ = hitters_df_clean.drop(['Salary', 'League', 'Division', 'NewLeague'], axis=1).astype('float64')

# Define the feature set X.
X = pd.concat([X_, dummies[['League_N', 'Division_W', 'NewLeague_N']]], axis=1)
```

+ Code + Text RAM Disk Editing

[8]
```
def processSubset(feature_set):
    # Fit model on feature_set and calculate RSS
    model = sm.OLS(y,X[list(feature_set)])
    regr = model.fit()
    RSS = ((regr.predict(X[list(feature_set)]) - y) ** 2).sum()
    return {"model":regr, "RSS":RSS}
```

```
def getBest(k):

    tic = time.time()

    results = []

    for combo in itertools.combinations(X.columns, k):
        results.append(processSubset(combo))

    # Wrap everything up in a nice dataframe
    models = pd.DataFrame(results)

    # Choose the model with the highest RSS
    best_model = models.loc[models['RSS'].argmin()]

    toc = time.time()
    print("Processed", models.shape[0], "models on", k, "predictors in", (toc-tic), "seconds.")

    # Return the best model, along with some other useful information about the model
    return best_model
```

+ Code + Text RAM Disk Editing

[10]
```
models_best = pd.DataFrame(columns=["RSS", "model"])

tic = time.time()
for i in range(1,8):
    models_best.loc[i] = getBest(i)

toc = time.time()
print("Total elapsed time:", (toc-tic), "seconds.")
```

```
Processed 18 models on 1 predictors in 0.0734550952911377 seconds.
Processed 153 models on 2 predictors in 0.4512465000152588 seconds.
Processed 816 models on 3 predictors in 2.2422337532043457 seconds.
Processed 3060 models on 4 predictors in 8.809548616409302 seconds.
Processed 8568 models on 5 predictors in 25.15835404396057 seconds.
Processed 18564 models on 6 predictors in 57.356990814208984 seconds.
Processed 31824 models on 7 predictors in 96.8684663772583 seconds.
Total elapsed time: 191.8530240058899 seconds.
```

[13]
```
models_best.loc[2, "model"].rsquared
```

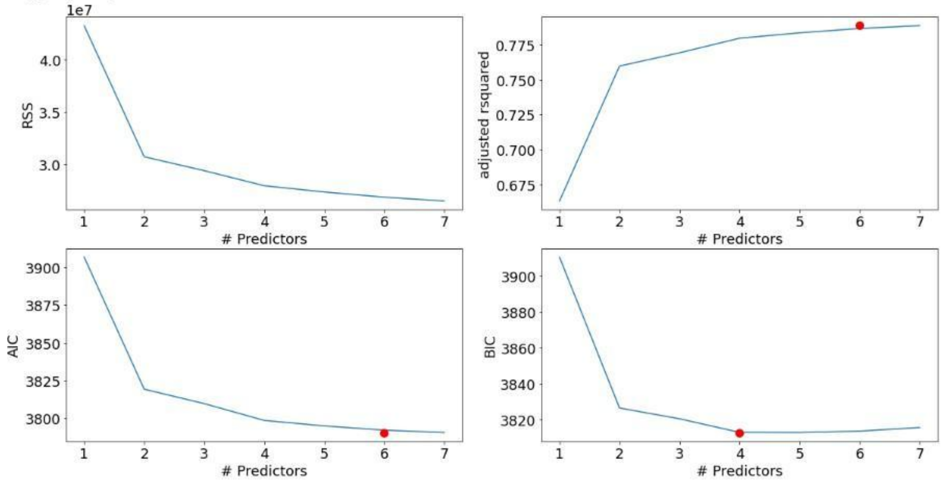
```
0.7614950002332872
```

```
[11] print(models_best.loc[2, "model"].summary())
```

```
                            OLS Regression Results
=======================================================================================
Dep. Variable:                 Salary   R-squared (uncentered):                   0.761
Model:                            OLS   Adj. R-squared (uncentered):              0.760
Method:                 Least Squares   F-statistic:                              416.7
Date:                Fri, 29 Apr 2022   Prob (F-statistic):                    5.80e-82
Time:                        10:46:05   Log-Likelihood:                         -1907.6
No. Observations:                 263   AIC:                                      3819.
Df Residuals:                     261   BIC:                                      3826.
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Hits           2.9538      0.261     11.335      0.000       2.441       3.467
CRBI           0.6788      0.066     10.295      0.000       0.549       0.809
==============================================================================
Omnibus:                      117.551   Durbin-Watson:                   1.933
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              654.612
Skew:                           1.729   Prob(JB):                    7.12e-143
Kurtosis:                       9.912   Cond. No.                         5.88
==============================================================================
```

```
plt.figure(figsize=(20,10))
plt.rcParams.update({'font.size': 18, 'lines.markersize': 10})

# Set up a 2x2 grid so we can look at 4 plots at once
plt.subplot(2, 2, 1)

# We will now plot a red dot to indicate the model with the largest adjusted R^2 statistic.
# The argmax() function can be used to identify the location of the maximum point of a vector
plt.plot(models_best["RSS"])
plt.xlabel('# Predictors')
plt.ylabel('RSS')

# We will now plot a red dot to indicate the model with the largest adjusted R^2 statistic.
# The argmax() function can be used to identify the location of the maximum point of a vector

rsquared_adj = models_best.apply(lambda row: row[1].rsquared_adj, axis=1)

plt.subplot(2, 2, 2)
plt.plot(rsquared_adj)
plt.plot(rsquared_adj.argmax(), rsquared_adj.max(), "or")
plt.xlabel('# Predictors')
plt.ylabel('adjusted rsquared')

# We'll do the same for AIC and BIC, this time looking for the models with the SMALLEST statistic
aic = models_best.apply(lambda row: row[1].aic, axis=1)

plt.subplot(2, 2, 3)
plt.plot(aic)
plt.plot(aic.argmin(), aic.min(), "or")
plt.xlabel('# Predictors')
plt.ylabel('AIC')

bic = models_best.apply(lambda row: row[1].bic, axis=1)

plt.subplot(2, 2, 4)
plt.plot(bic)
plt.plot(bic.argmin(), bic.min(), "or")
plt.xlabel('# Predictors')
plt.ylabel('BIC')
```

```
Text(0, 0.5, 'BIC')
```



```python
def forward(predictors):

    # Pull out predictors we still need to process
    remaining_predictors = [p for p in X.columns if p not in predictors]

    tic = time.time()

    results = []

    for p in remaining_predictors:
        results.append(processSubset(predictors+[p]))

    # Wrap everything up in a nice dataframe
    models = pd.DataFrame(results)

    # Choose the model with the highest RSS
    best_model = models.loc[models['RSS'].argmin()]

    toc = time.time()
    print("Processed ", models.shape[0], "models on", len(predictors)+1, "predictors in", (toc-tic), "seconds.")

    # Return the best model, along with some other useful information about the model
    return best_model
```

```python
models_fwd = pd.DataFrame(columns=["RSS", "model"])

tic = time.time()
predictors = []

for i in range(1,len(X.columns)+1):
    models_fwd.loc[i] = forward(predictors)
    predictors = models_fwd.loc[i]["model"].model.exog_names

toc = time.time()
print("Total elapsed time:", (toc-tic), "seconds.")
```

```
Processed  18 models on 1 predictors in 0.05907440185546875 seconds.
Processed  17 models on 2 predictors in 0.041628360748291016 seconds.
Processed  16 models on 3 predictors in 0.04813408851623535 seconds.
Processed  15 models on 4 predictors in 0.0421755313873291 seconds.
Processed  14 models on 5 predictors in 0.040349721908569336 seconds.
Processed  13 models on 6 predictors in 0.04169797897338867 seconds.
Processed  12 models on 7 predictors in 0.05890059471130371 seconds.
Processed  11 models on 8 predictors in 0.04334759712219238 seconds.
Processed  10 models on 9 predictors in 0.04883074760437012 seconds.
Processed  9 models on 10 predictors in 0.04139852523803711 seconds.
Processed  8 models on 11 predictors in 0.026102781295776367 seconds.
Processed  7 models on 12 predictors in 0.02393054962158203 seconds.
Processed  6 models on 13 predictors in 0.020375967025756836 seconds.
Processed  5 models on 14 predictors in 0.02061939239501953 seconds.
Processed  4 models on 15 predictors in 0.01975107192993164 seconds.
Processed  3 models on 16 predictors in 0.017058849334716797 seconds.
Processed  2 models on 17 predictors in 0.014175891876220703 seconds.
Processed  1 models on 18 predictors in 0.009147882461547852 seconds.
Total elapsed time: 0.7144672870635986 seconds.
```

+ Code + Text

RAM Disk Editing

```python
print(models_fwd.loc[1, "model"].summary())
print(models_fwd.loc[2, "model"].summary())
```

```
                                 OLS Regression Results
=======================================================================================
Dep. Variable:                 Salary   R-squared (uncentered):                   0.665
Model:                            OLS   Adj. R-squared (uncentered):              0.663
Method:                 Least Squares   F-statistic:                              519.2
Date:                Fri, 29 Apr 2022   Prob (F-statistic):                    4.20e-64
Time:                        10:59:34   Log-Likelihood:                         -1952.4
No. Observations:                 263   AIC:                                      3907.
Df Residuals:                     262   BIC:                                      3910.
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Hits           4.8833      0.214     22.787      0.000       4.461       5.305
==============================================================================
Omnibus:                       90.075   Durbin-Watson:                   1.949
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              293.080
Skew:                           1.469   Prob(JB):                     2.28e-64
Kurtosis:                       7.256   Cond. No.                         1.00
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
                                 OLS Regression Results
=======================================================================================
Dep. Variable:                 Salary   R-squared (uncentered):                   0.761
Model:                            OLS   Adj. R-squared (uncentered):              0.760
Method:                 Least Squares   F-statistic:                              416.7
Date:                Fri, 29 Apr 2022   Prob (F-statistic):                    5.80e-82
Time:                        10:59:34   Log-Likelihood:                         -1907.6
No. Observations:                 263   AIC:                                      3819.
Df Residuals:                     261   BIC:                                      3826.
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Hits           2.9538      0.261     11.335      0.000       2.441       3.467
CRBI           0.6788      0.066     10.295      0.000       0.549       0.809
==============================================================================
Omnibus:                      117.551   Durbin-Watson:                   1.933
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              654.612
Skew:                           1.729   Prob(JB):                    7.12e-143
Kurtosis:                       9.912   Cond. No.                         5.88
==============================================================================
```

```python
def backward(predictors):

    tic = time.time()

    results = []

    for combo in itertools.combinations(predictors, len(predictors)-1):
        results.append(processSubset(combo))

    # Wrap everything up in a nice dataframe
    models = pd.DataFrame(results)

    # Choose the model with the highest RSS
    best_model = models.loc[models['RSS'].argmin()]

    toc = time.time()
    print("Processed ", models.shape[0], "models on", len(predictors)-1, "predictors in", (toc-tic), "seconds.")

    # Return the best model, along with some other useful information about the model
    return best_model
```

```python
models_bwd = pd.DataFrame(columns=["RSS", "model"], index = range(1,len(X.columns)))

tic = time.time()
predictors = X.columns

while(len(predictors) > 1):
    models_bwd.loc[len(predictors)-1] = backward(predictors)
    predictors = models_bwd.loc[len(predictors)-1]["model"].model.exog_names

toc = time.time()
print("Total elapsed time:", (toc-tic), "seconds.")
```

```
Processed  18 models on 17 predictors in 0.10721325874328613 seconds.
Processed  17 models on 16 predictors in 0.11910152435302734 seconds.
Processed  16 models on 15 predictors in 0.10109567642211914 seconds.
Processed  15 models on 14 predictors in 0.09379887580871582 seconds.
Processed  14 models on 13 predictors in 0.07668757438659668 seconds.
Processed  13 models on 12 predictors in 0.07225203514099121 seconds.
Processed  12 models on 11 predictors in 0.07347846031188965 seconds.
Processed  11 models on 10 predictors in 0.06510496139526367 seconds.
Processed  10 models on 9 predictors in 0.05334353446960449 seconds.
Processed  9 models on 8 predictors in 0.04438591003417969 seconds.
Processed  8 models on 7 predictors in 0.038527488708496094 seconds.
Processed  7 models on 6 predictors in 0.04078793525695801 seconds.
Processed  6 models on 5 predictors in 0.031015872955322266 seconds.
Processed  5 models on 4 predictors in 0.04077720642089844 seconds.
Processed  4 models on 3 predictors in 0.02115154266357422 seconds.
```

```python
print("------------")
print("Best Subset:")
print("------------")
print(models_best.loc[7, "model"].params)
```

```
------------
Best Subset:
------------
AtBat         -0.989016
Hits           6.096745
CAtBat        -0.123657
CRuns          0.971103
CRBI           0.602747
PutOuts        0.309260
Division_W   -88.293555
dtype: float64
```

[22]
```python
print("-----------------")
print("Foward Selection:")
print("-----------------")
print(models_fwd.loc[7, "model"].params)
```

```
-----------------
Foward Selection:
-----------------
Hits           6.096745
CRBI           0.602747
Division_W   -88.293555
PutOuts        0.309260
AtBat         -0.989016
CRuns          0.971103
CAtBat        -0.123657
dtype: float64
```

[23]
```python
print("-------------------")
print("Backward Selection:")
print("-------------------")
print(models_bwd.loc[7, "model"].params)
```

```
-------------------
Backward Selection:
-------------------
AtBat         -0.989016
Hits           6.096745
CAtBat        -0.123657
CRuns          0.971103
CRBI           0.602747
PutOuts        0.309260
Division_W   -88.293555
dtype: float64
```

Questions:

1. Explain other data reduction techniques in brief.

Ans.

1. Dimensionality Reduction:

Whenever we encounter weakly important data, we use the attribute required for our analysis. Dimensionality reduction eliminates the attributes from the data set under consideration, thereby reducing the volume of original data. It reduces data size as it eliminates outdated or redundant features. Here are three methods of dimensionality reduction.

2. Numerosity Reduction:

The numerosity reduction reduces the original data volume and represents it in a much smaller form. This technique includes two types parametric and non parametric numerosity reduction.

3. Non-Parametric:

A non-parametric numerosity reduction technique does not assume any model. The non-Parametric technique results in a more uniform reduction, irrespective of data size, but it may not achieve a high volume of data reduction like the parametric. There are at least four types of Non-Parametric data reduction techniques, Histogram, Clustering, Sampling, Data Cube Aggregation, and Data Compression.

- **Histogram:** A histogram is a graph that represents frequency distribution which describes how often a value appears in the data. Histogram uses the binning method to represent an attribute's data distribution. It uses a disjoint subset which we call bin or buckets.

  A histogram can represent a dense, sparse, uniform, or skewed data. Instead of only one attribute, the histogram can be implemented for multiple attributes. It can effectively represent up to five attributes.
- **Clustering:** Clustering techniques groups similar objects from the data so that the objects in a cluster are similar to each other, but they are dissimilar to objects in another cluster.The quality of the cluster depends on the diameter of the cluster, i.e., the max distance between any two objects in the cluster.

  The cluster representation replaces the original data. This technique is more effective if the present data can be classified into a distinct clustered.

- **Sampling:** One of the methods used for data reduction is sampling, as it can reduce the large data set into a much smaller data sample. Below we will discuss the different methods in which we can sample a large data set D containing N tuples.

Outcomes:

CO3. Apply the transformations required on data to make it suitable for mining.

Conclusion: (Conclusion to be based on the objectives and outcomes achieved)
We successfully implemented data reduction of subsets.

Grade: AA / AB / BB / BC / CC / CD /DD

Signature of faculty in-charge with date

References:

Books/ Journals/ Websites:

1. Han, Kamber, "Data Mining Concepts and Techniques", Morgan Kaufmann 3rd Edition
2. Subset Selection is a Python adaptation of p. 244-247 of "Introduction to Statistical Learning with Applications in R" by Gareth James, Daniela Witten, Trevor Hastie and Robert Tibshirani. Adapted by R. Jordan Crouser at Smith College for SDS293: Machine Learning (Spring 2016).
3. Dataset: https://www.kaggle.com/code/omeryasirkucuk/salary-prediction-models-on-hitters-dataset/data?select=Hitters.csv