Sanket_Q2

December 10, 2022

0.1 Question 02

[ ]:
```python
import pandas as pd
from mlxtend.feature_selection import SequentialFeatureSelector as sfs
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from mlxtend.plotting import plot_sequential_feature_selection as plot_sfs
import matplotlib.pyplot as plt
import statsmodels.api as sm
from sklearn.linear_model import LassoCV, LassoLarsCV, LassoLarsIC
import numpy as np
from sklearn.metrics import r2_score, mean_squared_error
from math import sqrt
from scipy.stats import pearsonr
```

[ ]:
```python
data = pd.read_csv('College1.csv')
data.head()
```

[ ]:
```
                     Unnamed: 0 Private  Apps  Accept  Enroll  Top10perc  \
0  Abilene Christian University     Yes  1660    1232     721         23
1            Adelphi University     Yes  2186    1924     512         16
2                Adrian College     Yes  1428    1097     336         22
3           Agnes Scott College     Yes   417     349     137         60
4     Alaska Pacific University     Yes   193     146      55         16

   Top25perc  F.Undergrad  P.Undergrad  Outstate  Room.Board  Books  Personal  \
0         52         2885          537      7440        3300    450      2200
1         29         2683         1227     12280        6450    750      1500
2         50         1036           99     11250        3750    400      1165
3         89          510           63     12960        5450    450       875
4         44          249          869      7560        4120    800      1500

   PhD  Terminal  S.F.Ratio  perc.alumni  Expend  Grad.Rate
0   70        78       18.1           12    7041         60
1   29        30       12.2           16   10527         56
2   53        66       12.9           30    8735         54
3   92        97        7.7           37   19016         59
4   76        72       11.9            2   10922         15
```

[ ]:
```
data['Grad.Rate'].unique()
```

[ ]:
```
array([ 60,  56,  54,  59,  15,  55,  63,  73,  80,  52,  76,  74,  68,
        69, 100,  46,  34,  48,  70,  65,  88,  58,  71,  85,  79,  91,
        72,  84,  49,  82,  35,  51,  75,  53,  96,  67,  18,  33,  97,
        89,  93,  78,  83,  61,  81,  64,  62, 118,  24,  66,  47,  50,
        21,  87,  77,  43,  95,  37,  99,  45,  42,  98,  94,  38,  86,
        44,  22,  57,  29,  36,  39,  40,  26,  90,  92,  32,  27,  41,
        31,  10,  30], dtype=int64)
```

[ ]:
```
data.drop(['Unnamed: 0', 'Private'], axis=1, inplace=True)
```

[ ]:
```
X = data.drop('Outstate', axis=1)
Y = data['Outstate']
```

[ ]:
```
X.head(2)
```

[ ]:
```
(777,)
```

0.2 Part A

[ ]:
```
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.2,␣
 ↪random_state=42)
x_train.shape, x_test.shape
```

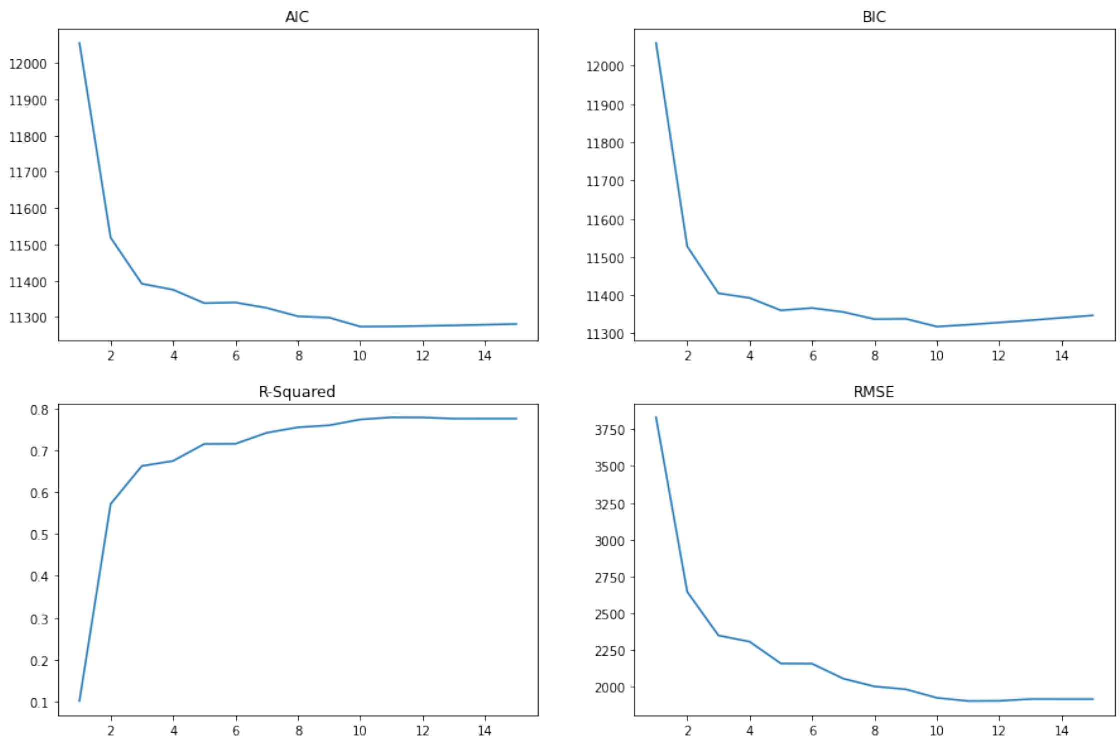
[ ]:
```
((621, 16), (156, 16))
```

0.3 Part A

[ ]:
```
aic = []
bic = []
r2 = []
rmse = []
nff = []

for i in range(1, 16): ## number of columns

    features_selector_forward = sfs(LinearRegression(),
                k_features=i,
                forward=True,
                verbose=0,
                cv=5, n_jobs=-1 )

    features_selector_forward.fit(x_train, y_train)

    mod = sm.GLS(y_train, x_train[list(features_selector_forward.
 ↪k_feature_names_)]).fit()
    y_pred = mod.predict(x_test[list(features_selector_forward.
 ↪k_feature_names_)])
```

```python
    mod_aic = mod.aic
    mod_bic = mod.bic
    mod_r2 = r2_score(y_test, y_pred)
    mod_rmse = sqrt(mean_squared_error(y_test, y_pred))

    aic.append(mod_aic)
    bic.append(mod_bic)
    r2.append(mod_r2)
    rmse.append(mod_rmse)
    nff.append(i)
```

[ ]:
```python
plt.figure(figsize=(15,10))

plt.subplot(2, 2, 1)
plt.plot(nff, aic)
plt.title("AIC")

plt.subplot(2, 2, 2)
plt.plot(nff, bic)
plt.title("BIC")

plt.subplot(2, 2, 3)
plt.plot(nff, r2)
plt.title("R-Squared")

plt.subplot(2, 2, 4)
plt.plot(nff, rmse)
plt.title("RMSE")

plt.show()
```



0.4

A lower AIC and BIC scores means the model is fitting good.
The possible answer for the number of good parametrs is 10

0.5 Part B

`-- Lets train the model with taking just the 10 parameters`

[ ]:
```
features_selector_forward = sfs(LinearRegression(),
                    k_features=10,
                    forward=True,
                    verbose=0,
                    cv=5, n_jobs=-1 )

features_selector_forward.fit(x_train, y_train)
```

[ ]:
```
SequentialFeatureSelector(estimator=LinearRegression(), k_features=10,
                          n_jobs=-1)
```

[ ]:
```
x_train = x_train[list(features_selector_forward.k_feature_names_)]
x_test = x_test[list(features_selector_forward.k_feature_names_)]
```

[ ]:
```
from pygam import LinearGAM
gam_model = LinearGAM().gridsearch(x_train.values, y_train.values)
```

```
  0% (0 of 11) |                        | Elapsed Time: 0:00:00 ETA:  --:--:--
  9% (1 of 11) |##                      | Elapsed Time: 0:00:00 ETA:   0:00:02
 18% (2 of 11) |####                    | Elapsed Time: 0:00:00 ETA:   0:00:02
 27% (3 of 11) |######                  | Elapsed Time: 0:00:00 ETA:   0:00:02
 36% (4 of 11) |#########               | Elapsed Time: 0:00:01 ETA:   0:00:01
 45% (5 of 11) |###########             | Elapsed Time: 0:00:01 ETA:   0:00:01
 54% (6 of 11) |#############           | Elapsed Time: 0:00:01 ETA:   0:00:01
 63% (7 of 11) |###############         | Elapsed Time: 0:00:02 ETA:   0:00:01
 72% (8 of 11) |##################      | Elapsed Time: 0:00:02 ETA:   0:00:00
 81% (9 of 11) |####################    | Elapsed Time: 0:00:02 ETA:   0:00:00
 90% (10 of 11) |#####################   | Elapsed Time: 0:00:02 ETA:   0:00:00
100% (11 of 11) |########################| Elapsed Time: 0:00:03 Time:  0:00:03
```

[ ]:
```
gam_model.summary()
```

```
LinearGAM
===============================================
==========================================================
Distribution:                        NormalDist Effective DoF:
37.091
Link Function:                     IdentityLink Log Likelihood:
-9942.4488
Number of Samples:                          621 AIC:
19961.0796
                                                AICc:

19966.1973
                                                GCV:

4010817.7966
                                                Scale:

3582049.0087
                                                Pseudo R-Squared:

0.791
================================================================================
==========================
Feature Function                  Lambda               Rank         EDoF
P > x        Sig. Code
================================= ==================== ============ ============
============ ============
s(0)                              [63.0957]            20           5.9
8.38e-03     **
s(1)                              [63.0957]            20           2.7
3.19e-05     ***
s(2)                              [63.0957]            20           4.7
3.03e-01
s(3)                              [63.0957]            20           3.1
2.05e-13     ***
s(4)                              [63.0957]            20           4.2
```

```
1.02e-12     ***
s(5)                         [63.0957]           20           4.5
2.03e-01
s(6)                         [63.0957]           20           3.5
2.45e-01
s(7)                         [63.0957]           20           3.7
1.72e-02     *
s(8)                         [63.0957]           20           1.9
1.11e-16     ***
s(9)                         [63.0957]           20           3.0
6.99e-04     ***
intercept                                        1            0.0
1.11e-16     ***
================================================================================
==========================
Significance codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

WARNING: Fitting splines and a linear function to a feature introduces a model
identifiability problem
         which can cause p-values to appear significant when they are not.

WARNING: p-values calculated in this manner behave correctly for un-penalized
models or models with
         known smoothing parameters, but when smoothing parameters have been
estimated, the p-values
         are typically lower than they should be, meaning that the tests reject
the null too readily.

c:\Users\Shankii\anaconda3\envs\gputest\lib\site-
packages\ipykernel_launcher.py:1: UserWarning: KNOWN BUG: p-values computed in
this summary are likely much smaller than they should be.

Please do not make inferences based on these values!

Collaborate on a solution, and stay up to date at:
github.com/dswah/pyGAM/issues/163

  """Entry point for launching an IPython kernel.
```

0.6 Part C

[ ]:
```python
y_pred = gam_model.predict(x_test.values)
print("RMSE = ", sqrt(mean_squared_error(y_test, y_pred)))
print(gam_model.statistics_['AIC'], gam_model.
 ↪statistics_['pseudo_r2']['explained_deviance'])
print("RSS = ", np.sum(np.square(y_pred - y_test)))
```

```
RMSE =  1803.0139370231386
```

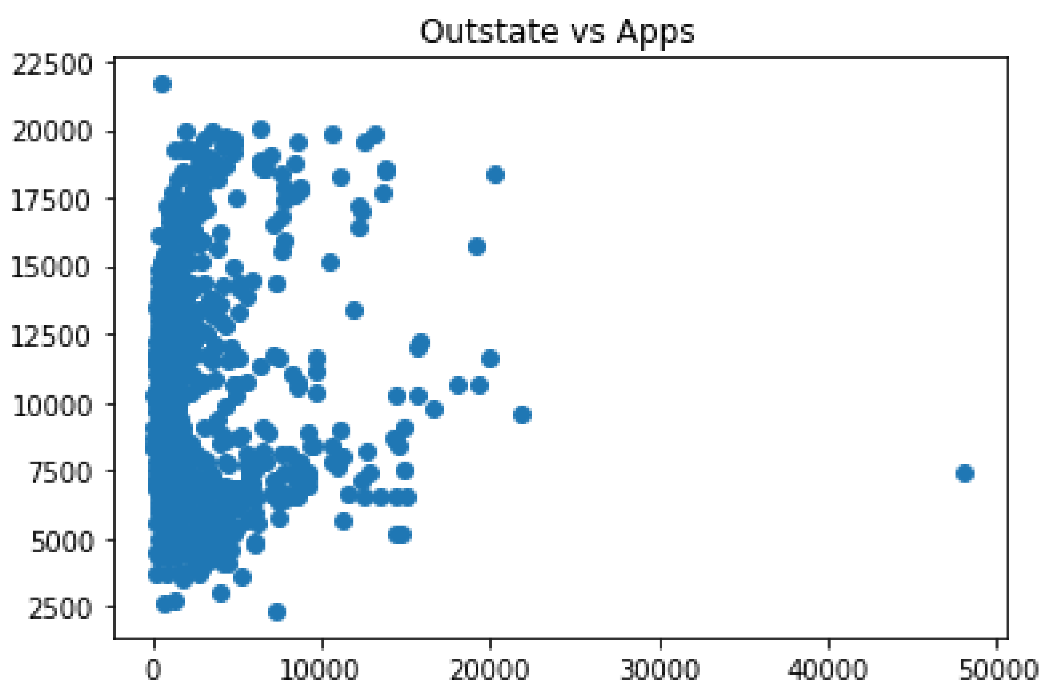
```
19961.0796450532 0.7910181713559215
RSS =  507134044.1075499
```

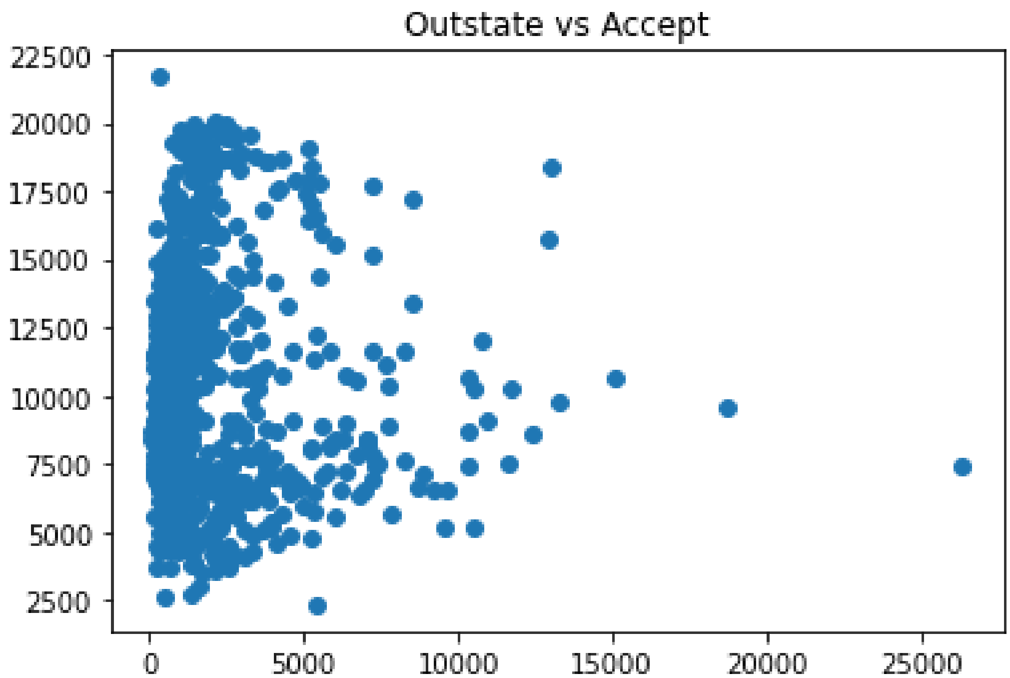
0.7 Part D

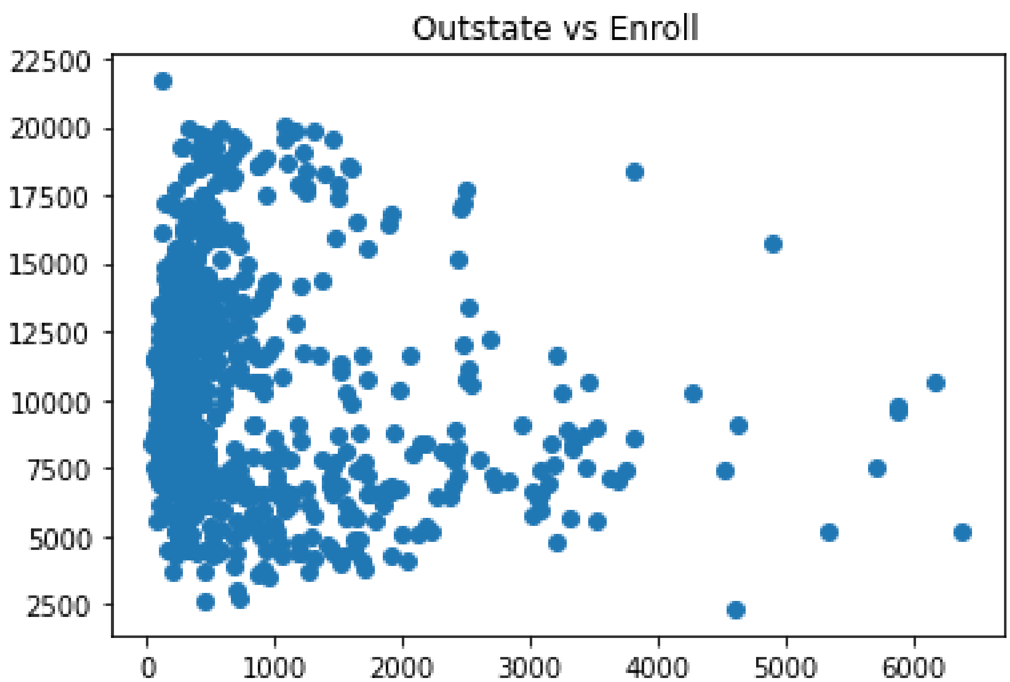
[ ]:
```
X.columns.shape
```

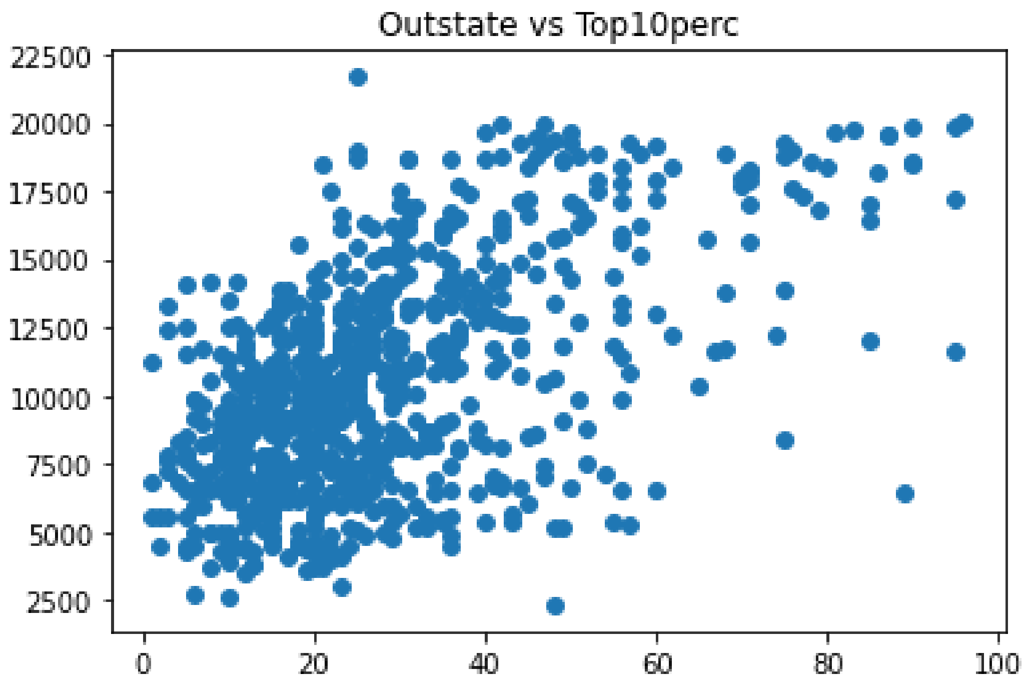
[ ]:
```
(16,)
```

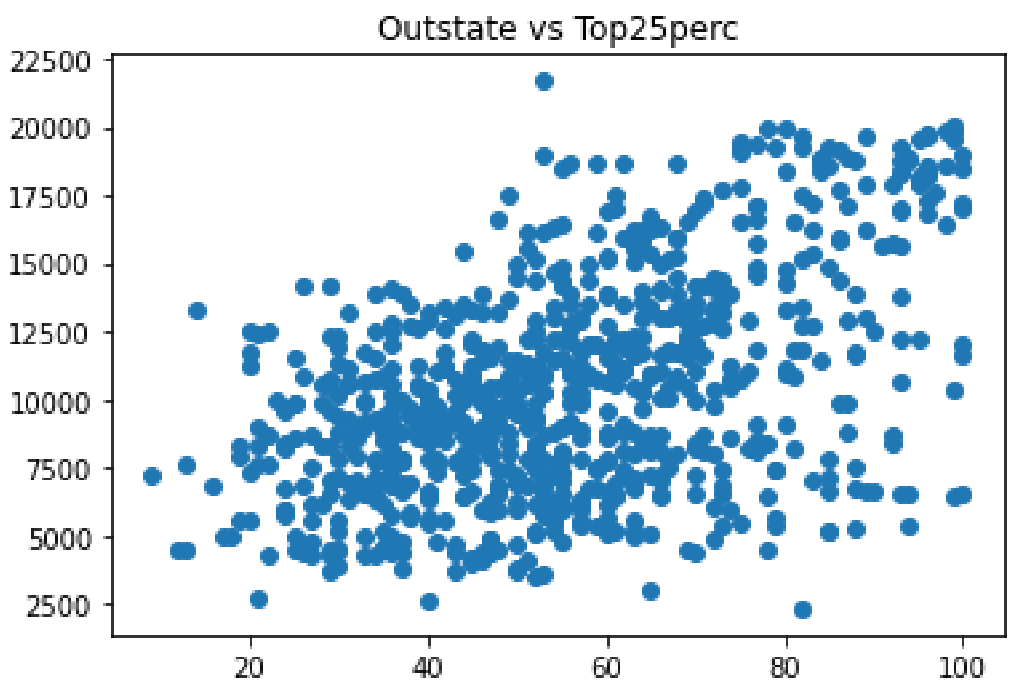
[ ]:
```
for i in X.columns:
    plt.scatter(X[i], Y)
    plt.title("{} vs {}".format('Outstate',i))
    plt.show()
```

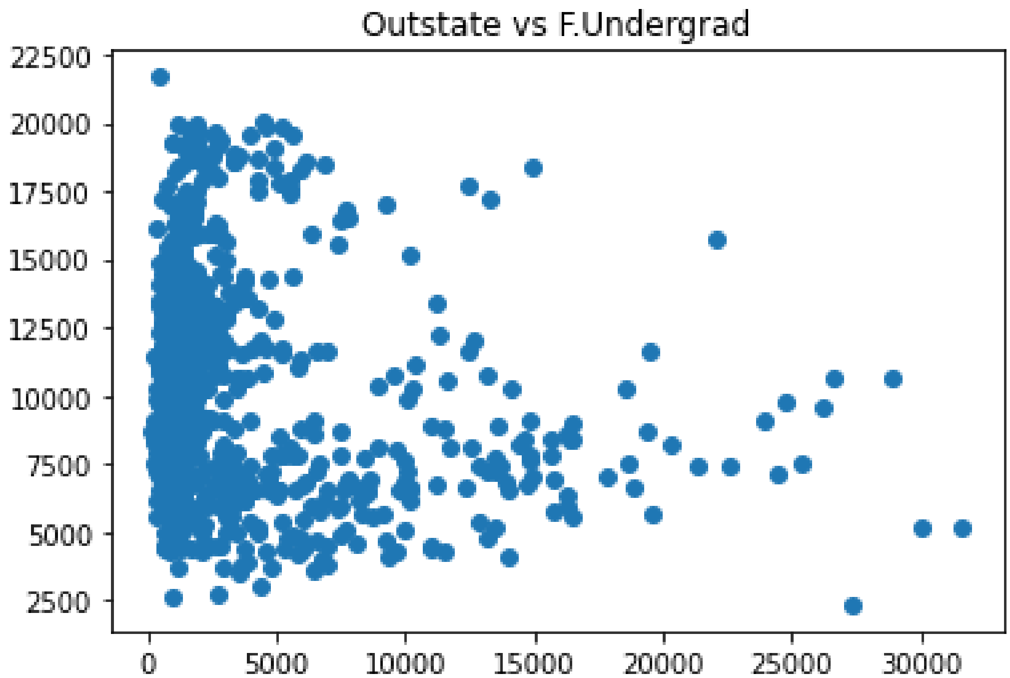


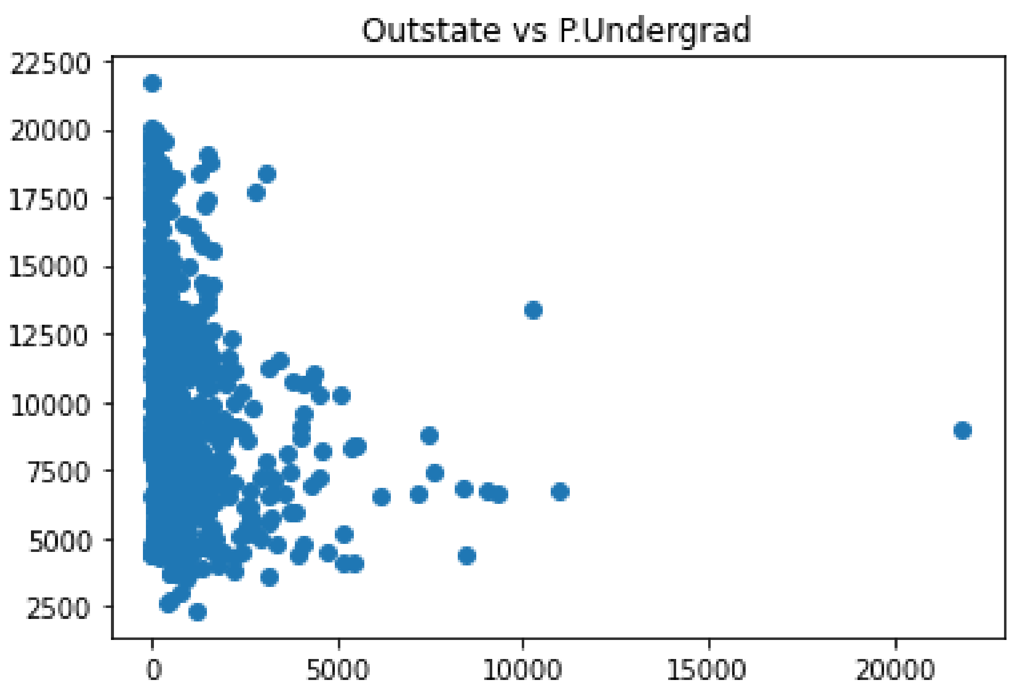

Outstate vs Accept
22500
20000
17500
15000
12500
10000
7500
5000
2500
0
5000
10000
15000
20000
25000

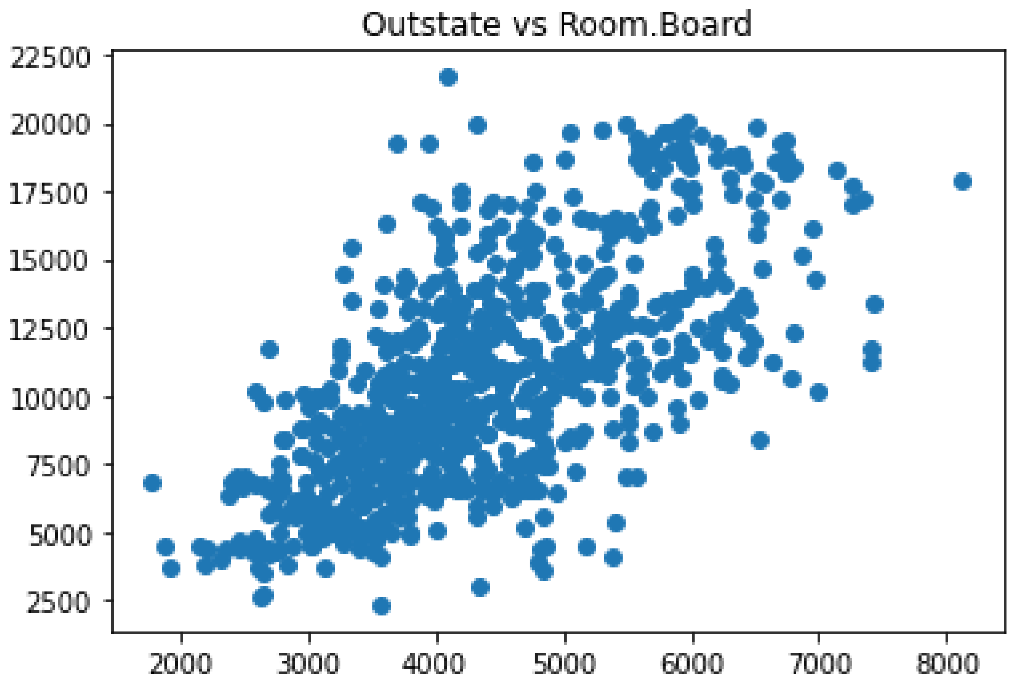

Outstate vs Enroll
22500
20000
17500
15000
12500
10000
7500
5000
2500
0
1000
2000
3000
4000
5000
6000


Outstate vs Top10perc
22500
20000
17500
15000
12500
10000
7500
5000
2500
0
20
40
60
80
100

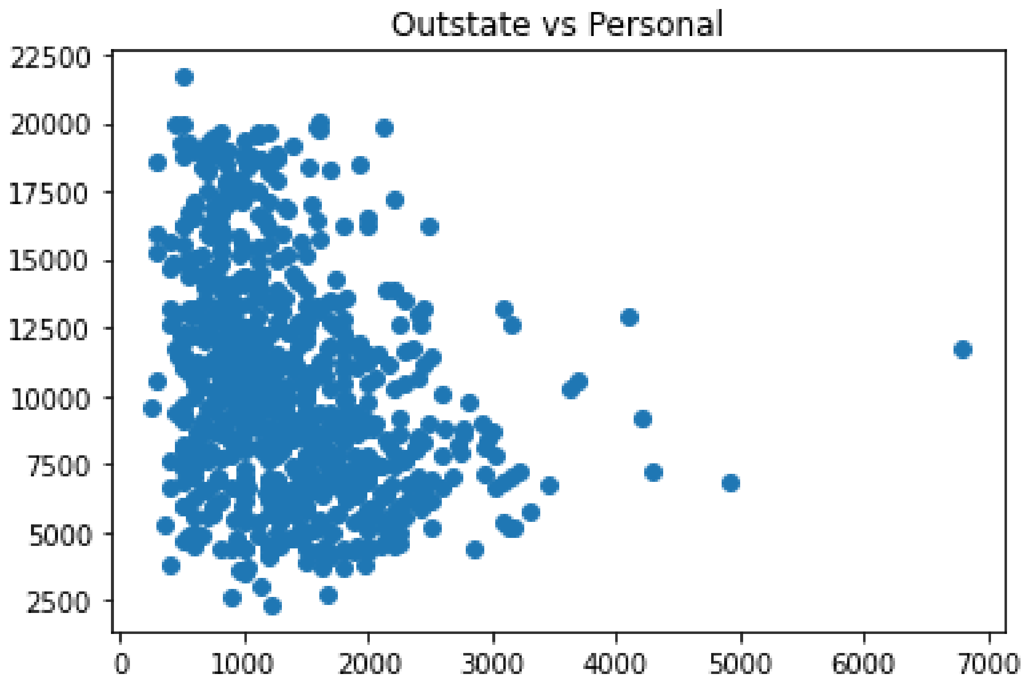

Outstate vs Top25perc
22500
20000
17500
15000
12500
10000
7500
5000
2500
20
40
60
80
100

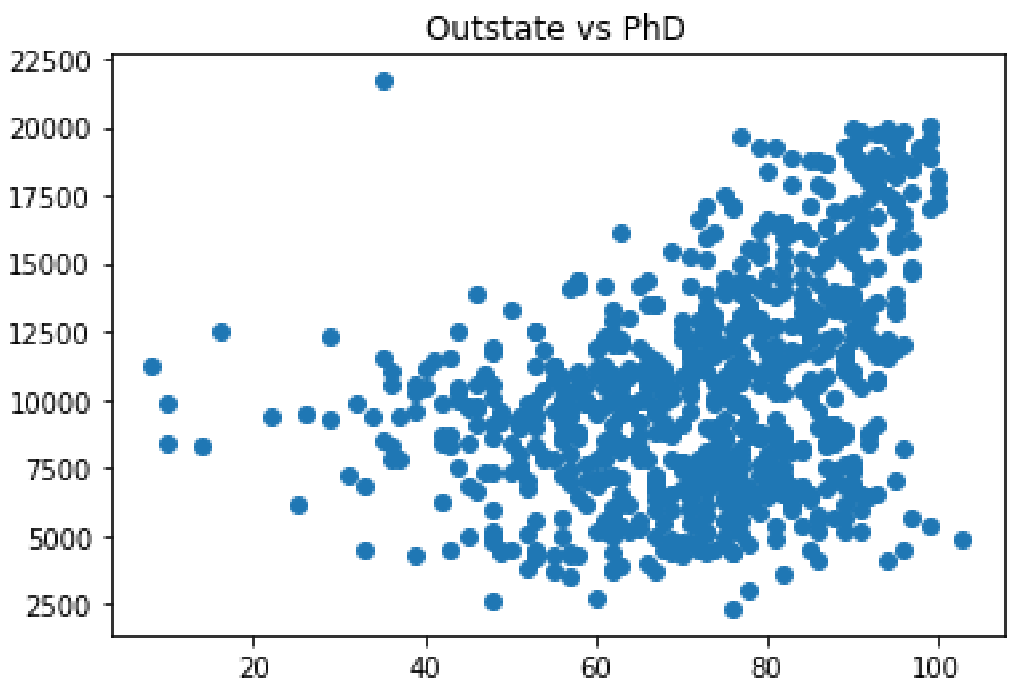

Outstate vs F.Undergrad
22500
20000
17500
15000
12500
10000
7500
5000
2500
0
5000
10000
15000
20000
25000
30000

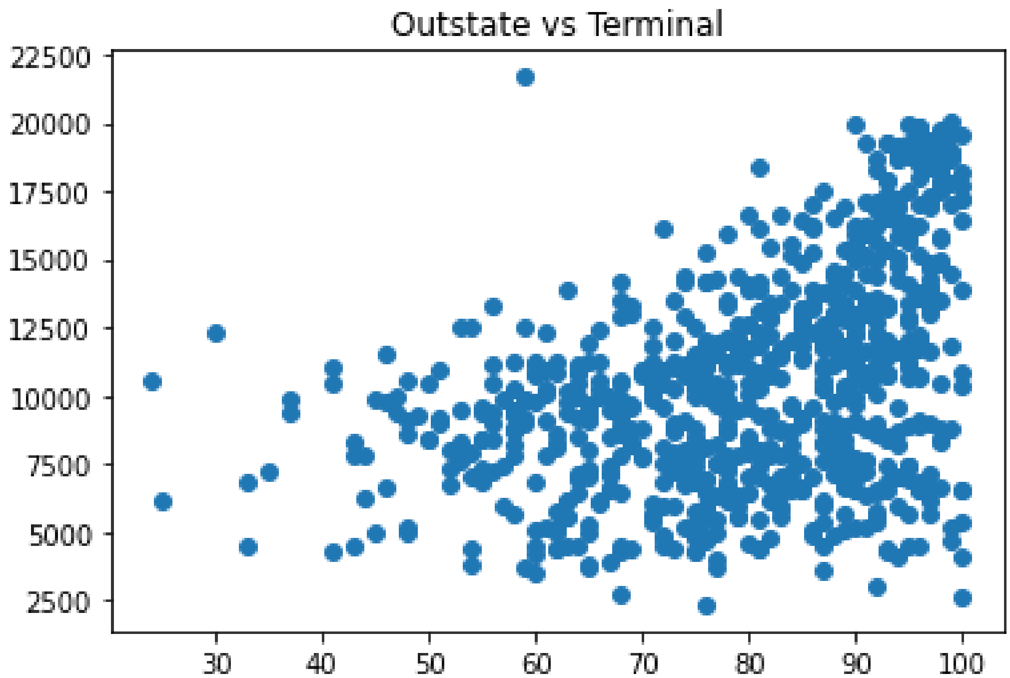

Outstate vs P.Undergrad
22500
20000
17500
15000
12500
10000
7500
5000
2500
0
5000
10000
15000
20000

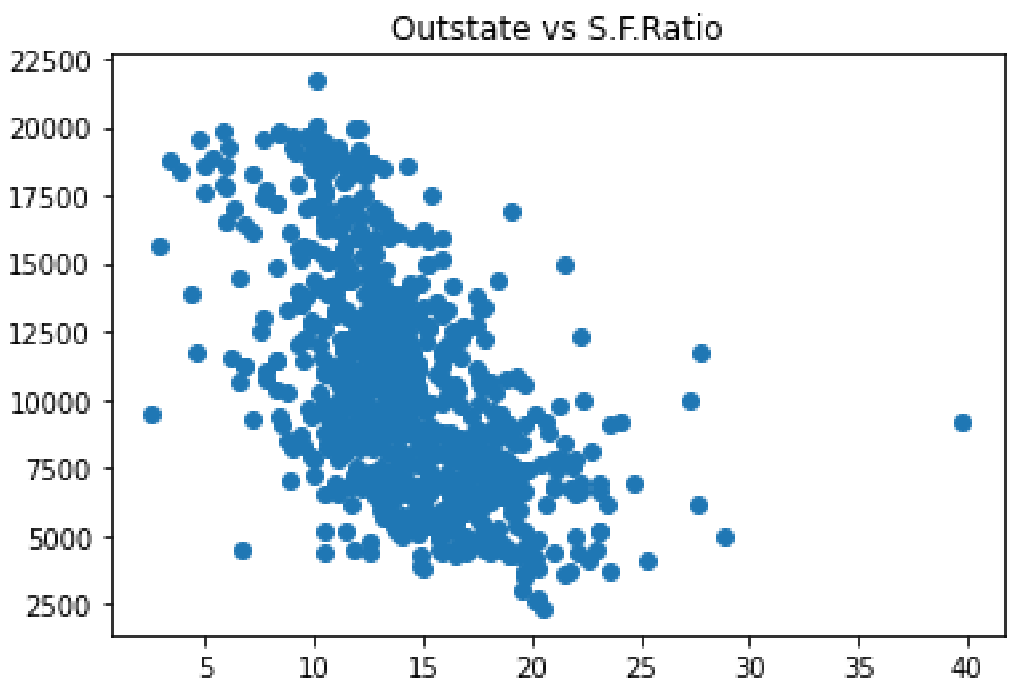

Outstate vs Room.Board
22500
20000
17500
15000
12500
10000
7500
5000
2500
2000
3000
4000
5000
6000
7000
8000

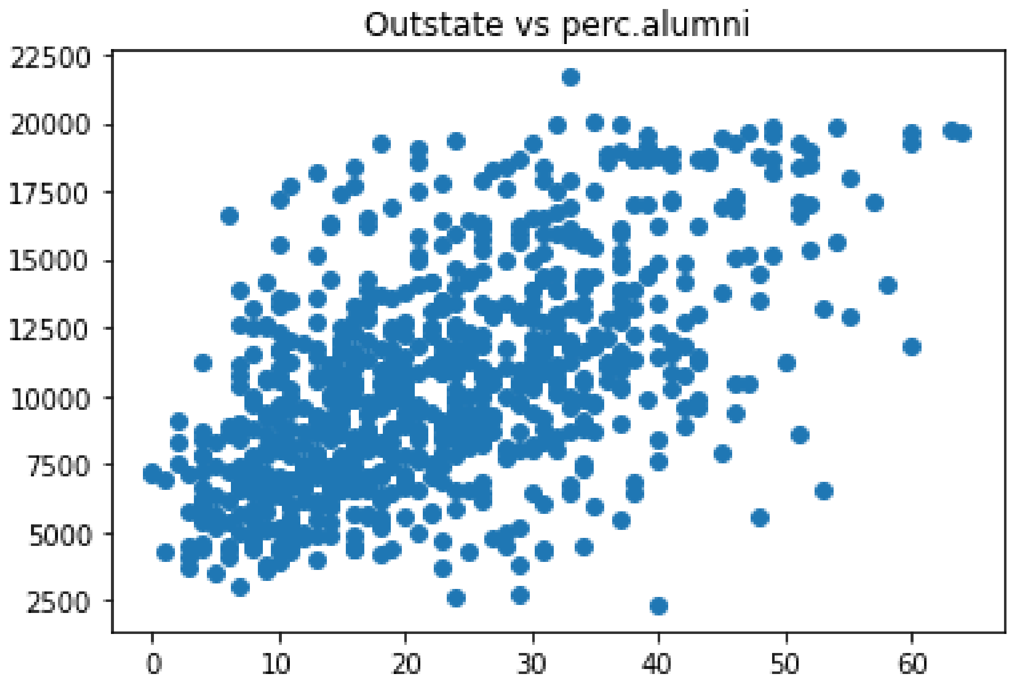

Outstate vs Books
22500
20000
17500
15000
12500
10000
7500
5000
2500
0
500
1000
1500
2000

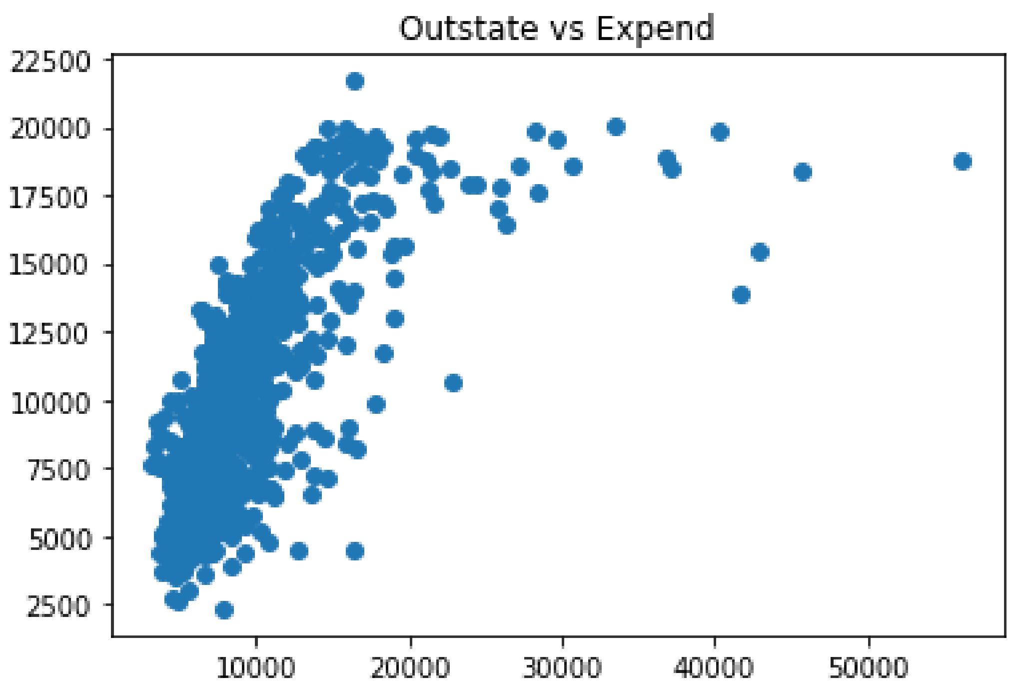

Outstate vs Personal
22500
20000
17500
15000
12500
10000
7500
5000
2500
0
1000
2000
3000
4000
5000
6000
7000

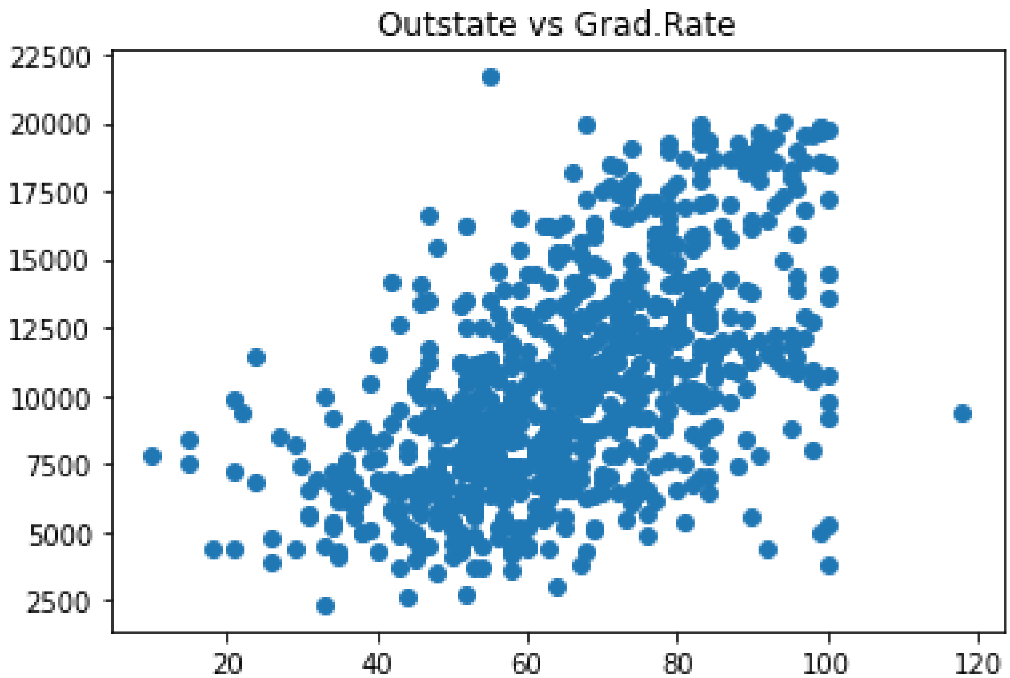

Outstate vs PhD
22500
20000
17500
15000
12500
10000
7500
5000
2500
20
40
60
80
100


Outstate vs Terminal
22500
20000
17500
15000
12500
10000
7500
5000
2500
30
40
50
60
70
80
90
100


Outstate vs S.F.Ratio
22500
20000
17500
15000
12500
10000
7500
5000
2500
5
10
15
20
25
30
35
40


Outstate vs perc.alumni
22500
20000
17500
15000
12500
10000
7500
5000
2500
0
10
20
30
40
50
60


Outstate vs Expend
22500
20000
17500
15000
12500
10000
7500
5000
2500
10000
20000
30000
40000
50000



– Can't really say which are linear and which are not
– Lets doo the Hypothesis testing:

```
Hypothesis Testing
H0 = The variables are non-linear
H1 = The variables are linear
```

[ ]:
```
for i in X[list(features_selector_forward.k_feature_names_)].columns:
    coeff, pvalue = pearsonr(X[i], Y)
    print("{} vs {}: pvalue = {}".format(i, "Outstate", pvalue))
```

```
Apps vs Outstate: pvalue = 0.16247452380683305
Accept vs Outstate: pvalue = 0.4734580801339919
Top10perc vs Outstate: pvalue = 5.459242638343674e-66
F.Undergrad vs Outstate: pvalue = 1.2340452116735829e-09
Room.Board vs Outstate: pvalue = 4.135091124156056e-96
Terminal vs Outstate: pvalue = 1.6021725306645543e-32
S.F.Ratio vs Outstate: pvalue = 6.23752219753576e-64
perc.alumni vs Outstate: pvalue = 4.349522462688962e-67
Expend vs Outstate: pvalue = 1.6298913528819218e-103
Grad.Rate vs Outstate: pvalue = 1.628926959311534e-68
```

0.8 P value < alpha: Reject H0

-> we will reject H0 for the variables except for Apps and Accept
    -> since for those variables: p-value > alpha