دانشگاه اصفهان

دانشکده مهندسی کامپیوتر

درس داده‌کاوی

پروژه‌ی فصل‌های ۶ و ۷

نگین شمس

خرداد ۱۴۰۱

- **تحلیل داده‌ها**

با بررسی داده‌ها می‌توان دریافت که این مجموعه داده فاقد مقادیر missing value می‌باشد.

```
data.isna().sum()
```

```
Relative_Compactness         0
Surface_Area                 0
Wall_Area                    0
Roof_Area                    0
Overall_Height               0
Orientation                  0
Glazing_Area                 0
Glazing_Area_Distribution    0
Heating_Load                 0
Cooling_Load                 0
dtype: int64
```

هم‌چنین این مجموعه فاقد داده‌ی تکراری است.

```
# generate count statistics of duplicate entries
if len(data[data.duplicated()]) > 0:
    print("No. of duplicated entries: ", len(data[data.duplicated()]))
    print(data[data.duplicated(keep=False)].sort_values(by=list(data.columns)).head())
else:
    print("No duplicated entries found")
```

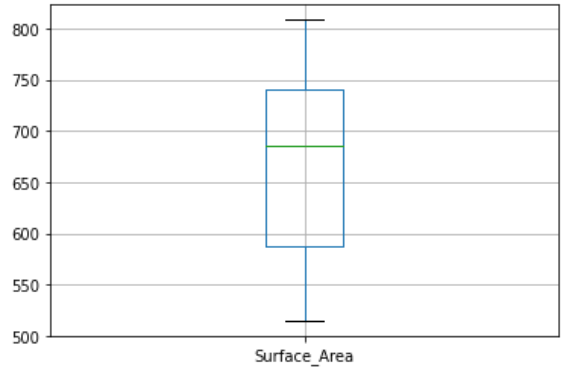
```
No duplicated entries found
```

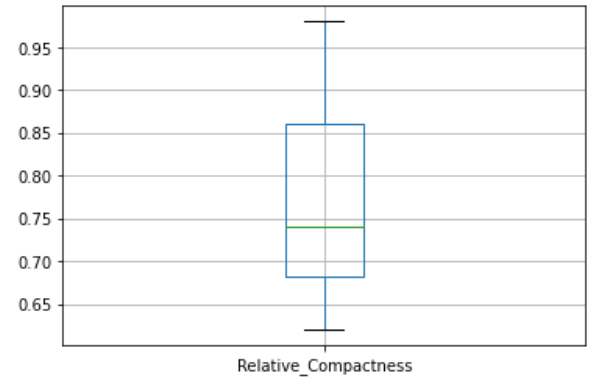
با ترسیم نمودار boxplot برای متغیرها، می‌توان چنین نتیجه گرفت که داده‌ها از توزیع نسبتا مناسبی برخوردار هستند و مشکل داده‌های پرت در مجموعه وجود ندارد. هم‌چنین با استفاده از تابع describe و مقایسه مقادیر min و max هر ستون با میانگین آن ستون، می‌توان تا حدودی به وجود یا عدم وجود داده‌های پرت پی برد.

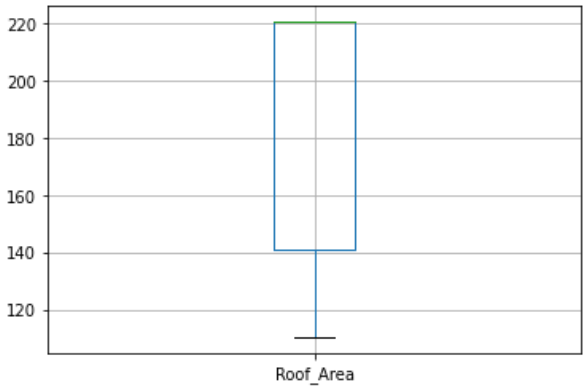
```
for c in data.columns:
  data.boxplot(column= c)
  plt.show()
```

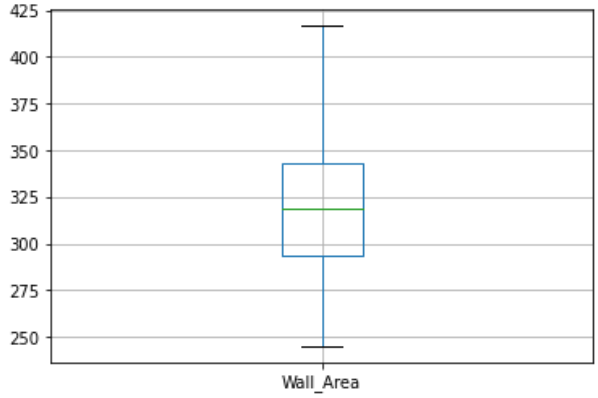
```
data.describe()
```

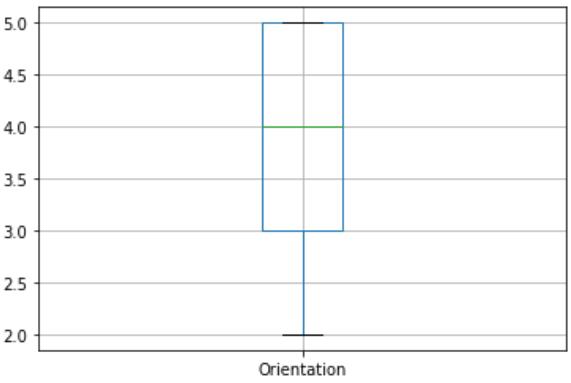
	Relative_Compactness	Surface_Area	Wall_Area	Roof_Area	Overall_Height	Orientation	Glazing_Area	Glazing_Area_Distribution	Heating_Load	Cooling_Load
count	668.000000	668.000000	668.000000	668.000000	668.000000	668.000000	668.000000	668.000000	668.000000	668.000000
mean	0.763338	672.502994	318.573353	176.964820	5.234281	3.526946	0.236377	2.797904	22.224461	24.494731
std	0.105762	88.245645	43.781674	45.180399	1.751241	1.110476	0.134583	1.551571	10.051225	9.483474
min	0.620000	514.500000	245.000000	110.250000	3.500000	2.000000	0.000000	0.000000	6.010000	10.900000
25%	0.682500	588.000000	294.000000	140.875000	3.500000	3.000000	0.100000	1.000000	13.000000	15.640000
50%	0.740000	686.000000	318.500000	220.500000	3.500000	4.000000	0.250000	3.000000	18.595000	21.700000
75%	0.860000	741.125000	343.000000	220.500000	7.000000	5.000000	0.400000	4.000000	31.555000	32.952500
max	0.980000	808.500000	416.500000	220.500000	7.000000	5.000000	0.400000	5.000000	43.100000	48.030000

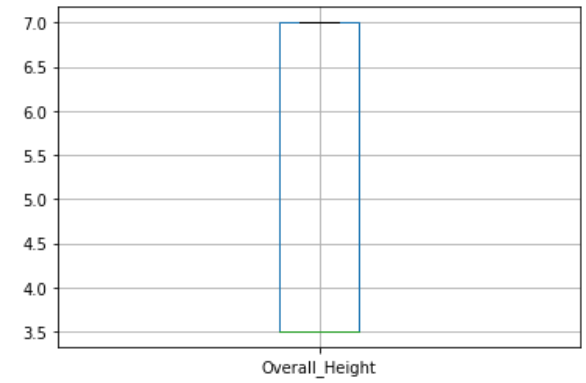

800
750
700
650
600
550
500
Surface_Area

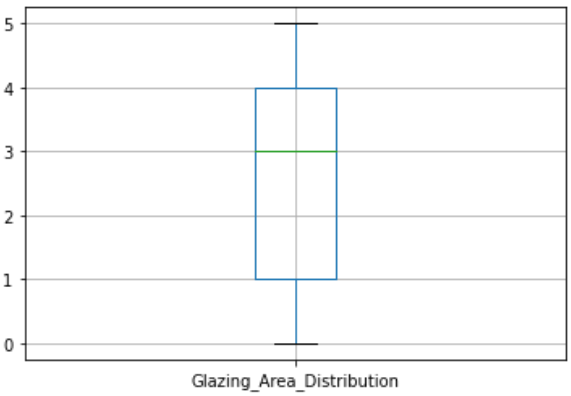

0.95
0.90
0.85
0.80
0.75
0.70
0.65
Relative_Compactness

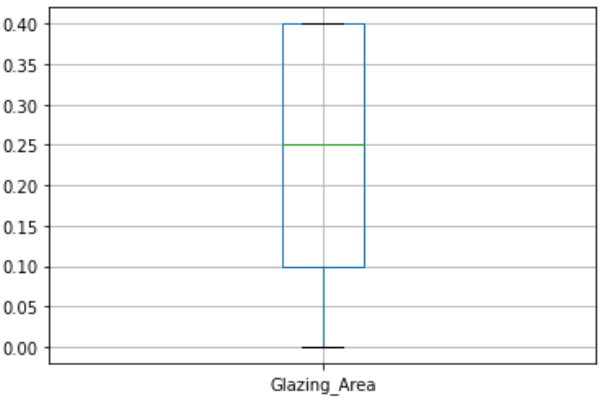

220
200
180
160
140
120
Roof_Area

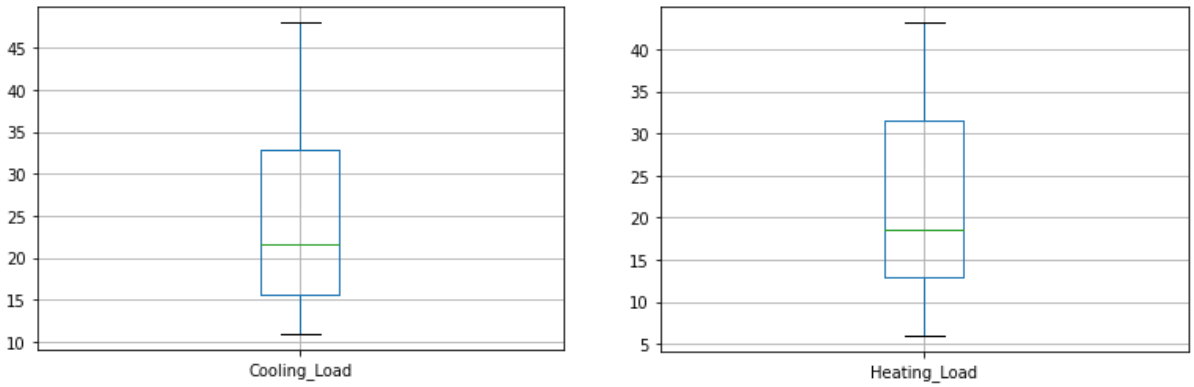

425
400
375
350
325
300
275
250
Wall_Area


5.0
4.5
4.0
3.5
3.0
2.5
2.0
Orientation


7.0
6.5
6.0
5.5
5.0
4.5
4.0
3.5
Overall_Height


5
4
3
2
1
0
Glazing_Area_Distribution

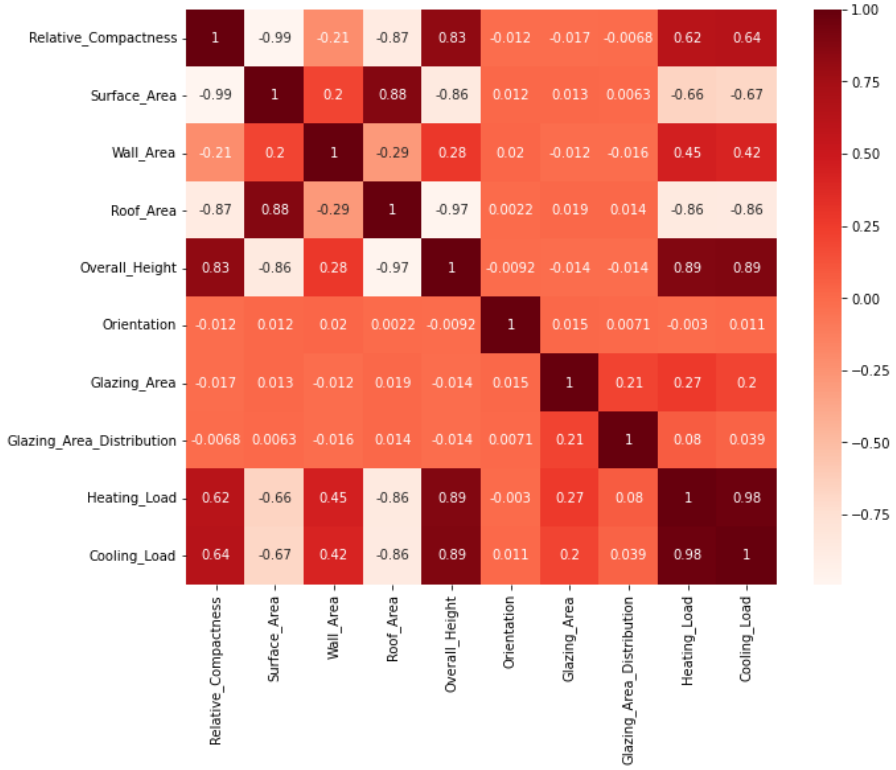

0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
Glazing_Area



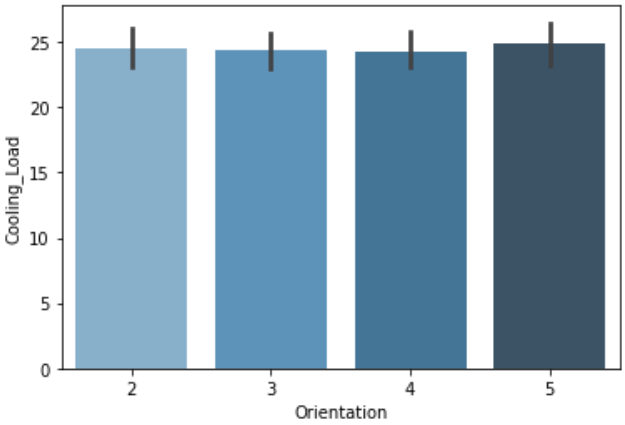
هم‌چنین با ترسیم نمودار با ترسیم نمودار heatmap می‌توان همبستگی بین متغیرهای مختلف را نمایش داد.

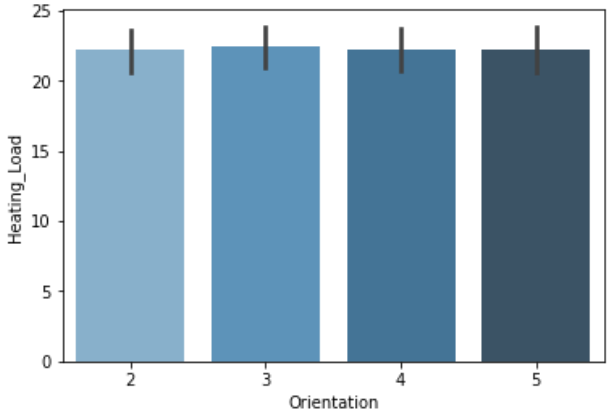
```
import seaborn as sns

dfCorr = data.corr()
plt.figure(figsize=(10,8))
sns.heatmap(dfCorr, annot=True, cmap="Reds")
plt.show()
```

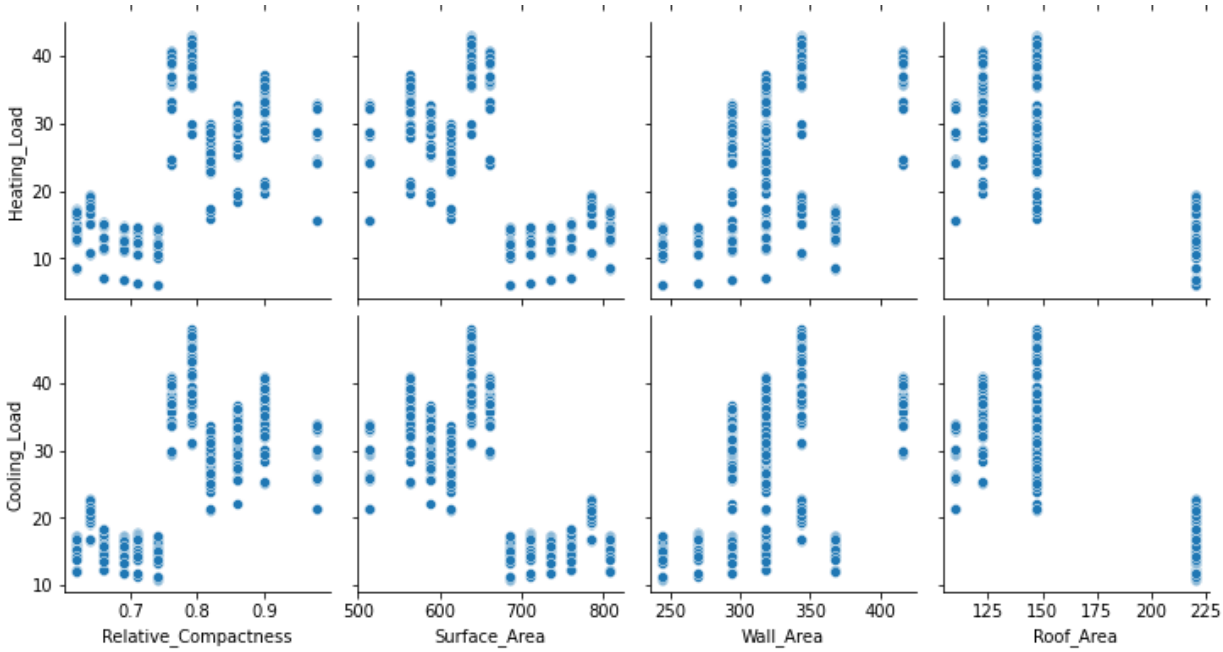


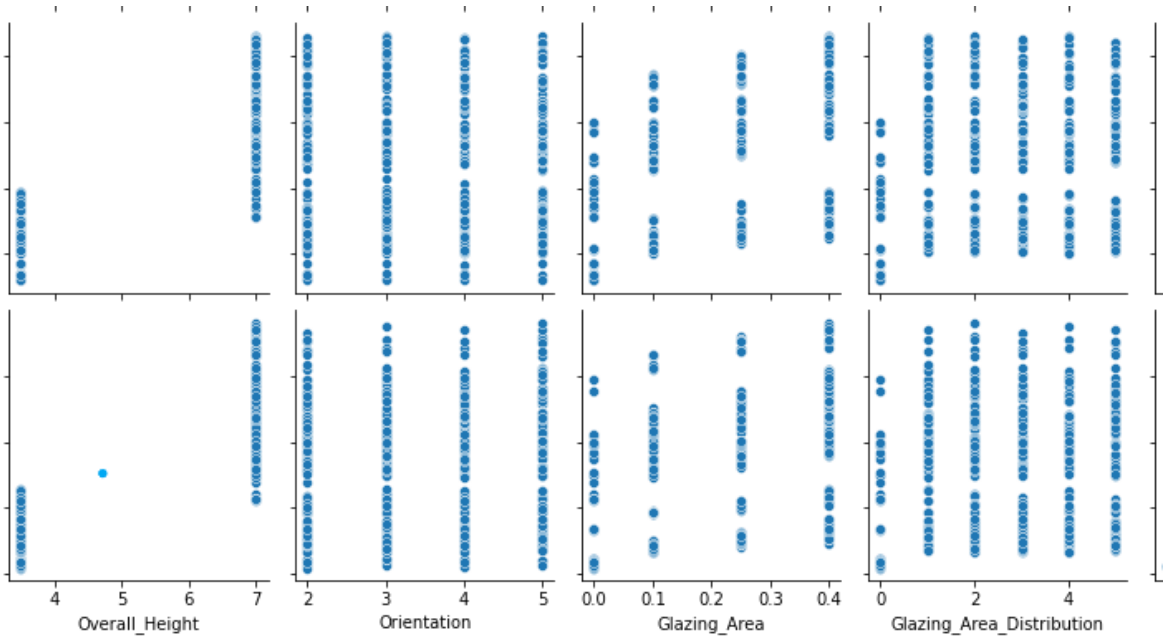
با استفاده از نمودار barplot می‌توان به این نتیجه رسید که مقادیر متفاوت ویژگی orientation تاثیر چندانی بر مقادیر ویژگی‌های Cooling-Load و Heating-Load ندارند. بنابراین می‌توان در صورت لزوم آن‌ها را حذف نمود.





نمودار پراکندگی داده‌ها براساس دو ستون Cooling-Load و Heating-Load به‌صورت زیر می‌باشد.





- **پیش‌بینی Heating Load**

پیش از شروع آموزش مدل داده‌ها نرمال‌سازی می‌شوند. برای نرمال‌سازی داده‌ها از standard scaler استفاده شده است.

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(data)
```

```
StandardScaler()
```

```
features = data.columns.tolist()
features.remove('Heating_Load')
features.remove('Cooling_Load')

data[features] = scaler.fit_transform(data[features])

test_data[features] = scaler.fit_transform(test_data[features])
```

به‌منظور پیش‌بینی متغیر Heating Load لازم است مقادیر این ستون به دو مقدار بالاتر یا برابر ۱۸ و پایین‌تر از ۱۸ تبدیل شوند. به‌ازای مقادیر بالاتر یا برابر ۱۸ عبارت high و به‌ازای مقادیر کمتر از ۱۸ عبارت low قرار می‌گیرد.

converting Heating Load column

```
data["Heating_Load"] = np.where(data["Heating_Load"] >= 18.0 , 'high', 'low')

test_data["Heating_Load"] = np.where(test_data["Heating_Load"] >= 18.0 , 'high', 'low')
```

سپس می‌توان با استفاده از oneHot Encoder مقادیر این ستون را به مقادیر عدد تبدیل نمود.

```
from sklearn.preprocessing import OneHotEncoder

onehotencoder = OneHotEncoder()
data['Heating_Load'] = onehotencoder.fit_transform(data[['Heating_Load']]).toarray()
test_data['Heating_Load'] = onehotencoder.fit_transform(test_data[['Heating_Load']]).toarray()
```

❖ درخت تصمیم

به‌منظور استفاده از مدل Decision Tree از کتابخانه‌ی sklearn استفاده شده است.

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import plot_confusion_matrix
from sklearn.metrics import classification_report
from sklearn import metrics

clf = DecisionTreeClassifier()
x_train, y_train = data.iloc[:, :-2], data.iloc[:, [-2]]
x_test, y_test = test_data.iloc[:, :-2], test_data.iloc[:, [-2]]

clf.fit(x_train, y_train)
clf.score(x_test, y_test)
predictions=clf.predict(x_test)

print("Confusion Matrix : ")
print(metrics.confusion_matrix(y_test, predictions))
print("\n Prediction Accuracy : ",  \
      metrics.accuracy_score(y_test, predictions) )
print("classification report:")
print(metrics.classification_report(y_test, predictions))
```

مقادیر ماتریس آشفتگی به‌صورت زیر می‌باشد:

```
Confusion Matrix :
[[44  1]
 [ 0 55]]

 Prediction Accuracy :  0.99
classification report:
              precision    recall  f1-score   support

         0.0       1.00      0.98      0.99        45
         1.0       0.98      1.00      0.99        55

    accuracy                           0.99       100
   macro avg       0.99      0.99      0.99       100
weighted avg       0.99      0.99      0.99       100
```

با استفاده از ماتریس آشفتگی، مقادیر زیر قابل محاسبه است:

- Accuracy

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN} = \frac{44+55}{44+1+0+55} = \frac{99}{100} = 0.99$$

- Error rate

$$\text{Error rate} = \frac{FN+FN}{TP+TN+FP+FN} = \frac{1+0}{44+1+0+55} = \frac{1}{100} = 0.01$$

- Sensitivity

$$\text{Sensitivity} = \frac{TP}{TP+FN} = \frac{44}{44+0} = 1$$

- Specificity

$$\text{Specificity} = \frac{TN}{TN+FP} = \frac{55}{55+1} = 0.98$$

- Precision

$$\text{Precision} = \frac{TP}{TP+FP} = \frac{44}{44+1} = 0.977$$

- Recall

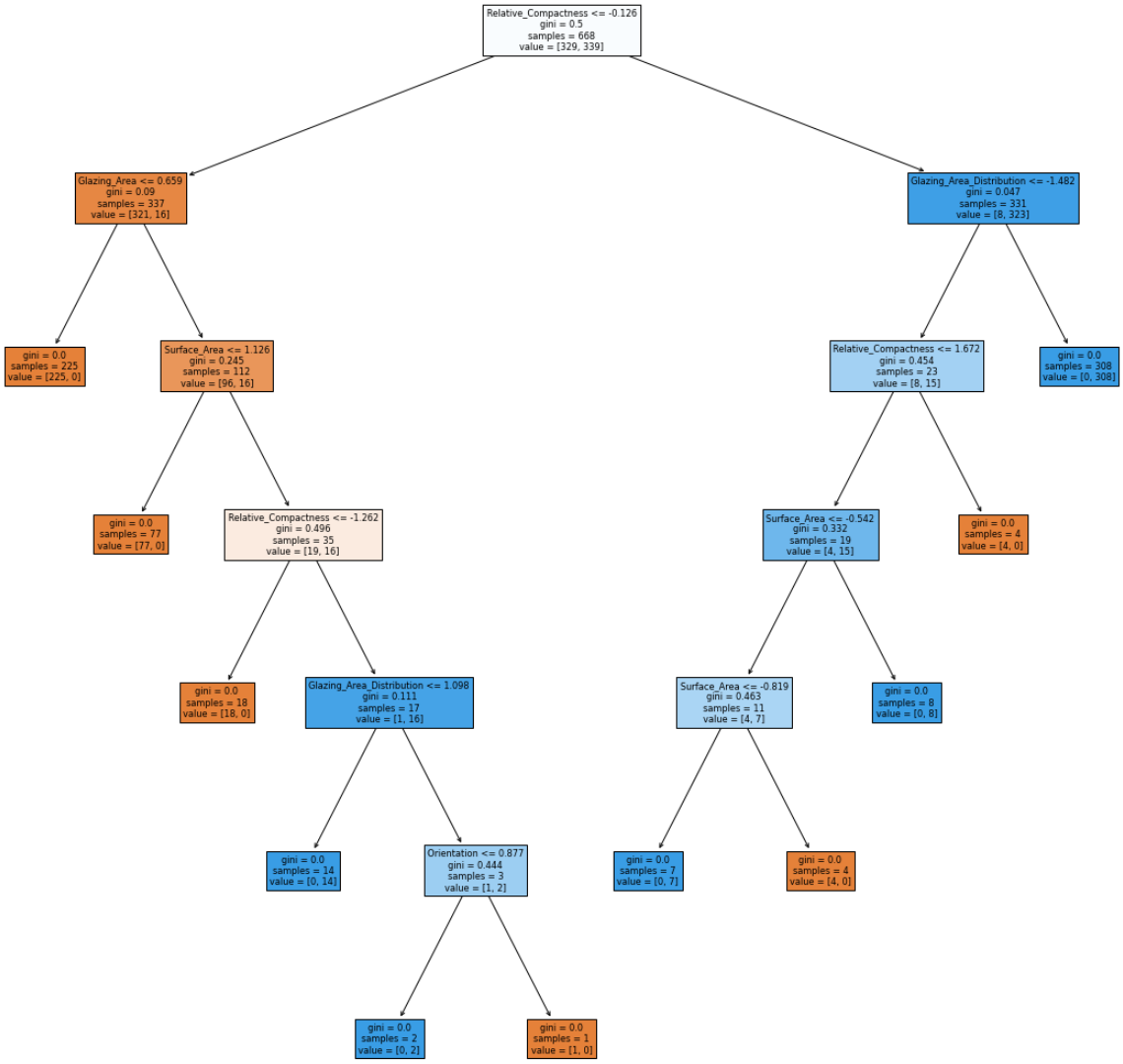
$$\text{Recall} = \frac{TN}{TN+FP} = \frac{55}{55+1} = 0.98$$

- F-score

$$\text{F-score} = \frac{2\times precision\times recall}{precision+recall} = \frac{2\times 0.977\times 0.98}{0.977+0.98} = 0.978$$

ساختار درخت تصمیم مذکور به‌صورت زیر می‌باشد:

```
from sklearn import tree
plt.figure(figsize = (20, 20))
tree.plot_tree(clf, feature_names = x_train.columns, filled= True)
plt.show()
```



❖ **Random Forest**

برای یافتن بهترین مقدار برای پارامترهای مدل Random Forest از روش gridsearch استفاده شده است. در این روش که بر مبنای cross validation می‌باشد، نتایج مدل به‌ازای پارامترهای مختلف به‌دست می‌آید و سپس پارامترهایی که بهترین نتیجه را داشته‌اند، به‌عنوان پارامترهای نهایی مدل انتخاب می‌شوند. بهترین مقدار به‌ازای پارامتر max_depth برابر ۸ و به‌ازای پارامتر n-estimator برابر ۶۴ می‌باشد.

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV

x_train, y_train = data.iloc[:, :-2], data.iloc[:, [-2]]
x_test, y_test = test_data.iloc[:, :-2], test_data.iloc[:, [-2]]

rf = RandomForestClassifier()
parameters = {
    'n_estimators': [2**i for i in range(3, 10)],
    'max_depth': [2, 4, 8, 16, 32, None]
}
cv = GridSearchCV(rf, parameters, cv=5)
cv.fit(x_train, y_train.values.ravel())

print('BEST PARAMS: {}\n'.format(cv.best_params_))

# print_results(cv)
```

```
BEST PARAMS: {'max_depth': 8, 'n_estimators': 64}
```

نتایج این مدل به‌صورت زیر می‌باشد:

```
from sklearn.metrics import confusion_matrix

y_pred = cv.best_estimator_.predict(x_test)
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
[[44  1]
 [ 0 55]]
              precision    recall  f1-score   support

         0.0       1.00      0.98      0.99        45
         1.0       0.98      1.00      0.99        55

    accuracy                           0.99       100
   macro avg       0.99      0.99      0.99       100
weighted avg       0.99      0.99      0.99       100
```

با توجه به نتایج به‌دست آمده، عملکرد این دو مدل و نتایج معیارهای ارزیابی برای این مجموعه داده‌ی تست، مشابه است.

- **پیش‌بینی Cooling Load**

```
preparing train and test sets
```

```
[ ]  y_train = data['Cooling_Load']
     x_train = data.drop(['Cooling_Load', 'Heating_Load'], axis = 1)

     y_test = test_data['Cooling_Load']
     x_test = test_data.drop(['Cooling_Load', 'Heating_Load'], axis = 1)
```

سپس مقادیر ستون Cooling Load به پنج دسته‌ی مختلف تقسیم می‌شود. برای این کار حد فاصل بین مقدار بیشینه و کمینه‌ی این ستون به پنج بازه با توزیع یکسان تقسیم می‌شود.

```
min_value = data['Cooling_Load'].min()
max_value = data['Cooling_Load'].max()

bins = np.linspace(min_value,max_value,6)
bins
```

```
array([10.9  , 18.326, 25.752, 33.178, 40.604, 48.03 ])
```

```
labels = ['level1', 'level2', 'level3', 'level4', 'level5']

data['Cooling_Load'] = pd.cut(data['Cooling_Load'], bins=bins, labels=labels, include_lowest=True)

test_data['Cooling_Load'] = pd.cut(test_data['Cooling_Load'], bins=bins, labels=labels, include_lowest=True)
```

❖ **SVM**

در صورتی‌که داده‌ها با استفاده از MinMax scaler نرمال‌سازی شوند، نتایج مدل SVM بهتر خواهد بود. به‌منظور استفاده از این طبقه‌بند برای مسائل چندکلاسه، لازم است از یک تابع کرنل استفاده شود. در ابتدا از تابع چندجمله‌ای استفاده شده است. برای مشخص نمودن مقدار درجه‌ی چند جمله‌ای، دقت مدل به‌ازای مقادیر ۲ تا ۱۲ درجه محاسبه شده است. با توجه به نتایج، بهترین گزینه تابع چندجمله‌ای درجه هفت می‌باشد زیرا بالاترین دقت یعنی ۸۷ درصد را می‌دهد.

```
from sklearn import svm

#Create a svm Classifier

for deg in range(2, 13):
  clf = svm.SVC(kernel='poly', degree = deg)

  #Train the model using the training sets
  clf.fit(x_train, y_train)

  #Predict the response for test dataset
  y_pred = clf.predict(x_test)

  print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.69
Accuracy: 0.7
Accuracy: 0.81
Accuracy: 0.85
Accuracy: 0.81
Accuracy: 0.87
Accuracy: 0.86
Accuracy: 0.85
Accuracy: 0.85
Accuracy: 0.85
Accuracy: 0.83
```

هم‌چنین دقت مدل به‌ازای توابع کرنل دیگر مانند تابع خطی، سیگموئید و rbf نیز محاسبه شده است اما دقت مدل در صورت استفاده از تابع کرنل درجه ۷ بیشتر می‌باشد.

```
kernels = ['linear', 'sigmoid', 'rbf', 'poly']
for kernel in kernels:
  clf = svm.SVC(kernel=kernel)
  clf.fit(x_train, y_train)
  y_pred = clf.predict(x_test)
  print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.68
Accuracy: 0.58
Accuracy: 0.7
Accuracy: 0.68
```

نتایج ارزیابی مدل SVM به‌صورت زیر می‌باشد. با توجه به این‌که در این حالت مسئله‌ی مورد نظر به‌صورت طبقه‌بندی چندکلاسه می‌باشد، بنابراین می‌توان از معیارهای macro و micro برای محاسبه‌ی precision، recall و F-score استفاده نمود. در روش macro، معیارهای ارزیابی مذکور به‌صورت مستقل برای هر کلاس محاسبه می‌شود. سپس میانگین این مقادیر به‌دست می‌آید. بنابراین، در این روش به همه‌ی کلاس‌ها ارزش یکسانی تعلق می‌گیرد. در روش micro، اندازه‌ی کلاس‌های مختلف نیز برای محاسبه میانگین در نظر گرفته می‌شود.

```
clf = svm.SVC(kernel='poly', degree = 7)
clf.fit(x_train, y_train)
y_pred = clf.predict(x_test)

print("Accuracy:", metrics.accuracy_score(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
Accuracy: 0.87
[[37  0  0  0  0]
 [ 0 12  2  0  0]
 [ 0  2 16  1  0]
 [ 0  1  5 17  2]
 [ 0  0  0  0  5]]
              precision    recall  f1-score   support

      level1       1.00      1.00      1.00        37
      level2       0.80      0.86      0.83        14
      level3       0.70      0.84      0.76        19
      level4       0.94      0.68      0.79        25
      level5       0.71      1.00      0.83         5

    accuracy                           0.87       100
   macro avg       0.83      0.88      0.84       100
weighted avg       0.89      0.87      0.87       100
```

❖ شبکه عصبی

با توجه به مقادیر ماتریس وابستگی، ویژگی orientation کم‌ترین همبستگی را با ویژگی Cooling-Load دارد (با ضریب همبستگی ۰.۰۱۱). این ویژگی در ادامه حذف می‌شود زیرا هر چه همبستگی یک ویژگی با ویژگی برچسب کمتر باشد، احتمالا در پیش‌بینی آن نقش کمتری دارد. با حذف این ویژگی دقت شبکه عصبی مورد استفاده افزایش می‌یابد.

برای آموزش شبکه عصبی نیز از grid search استفاده شده است. با توجه به نتایج بهترین مقدار به‌ازای پارامترهای مدل به‌صورت زیر می‌باشد:

- ✓ تعداد لایه‌های مخفی: ۵۰
- ✓ تعداد نورون‌های موجود در لایه‌های مخفی: ۱۰۰
- ✓ آلفا (نرخ یادگیری): ۰.۰۵.
- ✓ الگوریتم بهینه‌سازی (solver): adam
- ✓ تابع فعال‌سازی: relu

```
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import GridSearchCV

y_train = data['Cooling_Load']
x_train = data.drop(['Cooling_Load', 'Heating_Load'], axis = 1)

y_test = test_data['Cooling_Load']
x_test = test_data.drop(['Cooling_Load', 'Heating_Load'], axis = 1)

mlp = MLPClassifier(max_iter=100)

parameter_space = {
    'hidden_layer_sizes': [(5, 4),(3,5), (6,5), (8, 4), (10, 10), (10, 5), (20, 10), (50,50,50), (50,100,50), (100,)],
    'activation': ['tanh', 'relu', 'softmax'],
    'solver': ['sgd', 'adam'],
    'alpha': [0.0001, 0.05],
    'learning_rate': ['constant','adaptive'],
}

clf = GridSearchCV(mlp, parameter_space, n_jobs=-1, cv=3)
clf.fit(x_train, y_train)

# Best paramete set
print('Best parameters found:\n', clf.best_params_)
```

```
Best parameters found:
 {'activation': 'relu', 'alpha': 0.05, 'hidden_layer_sizes': (50, 100, 50), 'learning_rate': 'constant', 'solver': 'adam'}
```

نتایج حاصل از ارزیابی شبکه عصبی فوق به‌صورت زیر می‌باشد:

Results on the test set:

	precision	recall	f1-score	support
level1	1.00	1.00	1.00	37
level2	1.00	0.86	0.92	14
level3	0.68	0.79	0.73	19
level4	0.79	0.76	0.78	25
level5	0.80	0.80	0.80	5
accuracy			0.87	100
macro avg	0.85	0.84	0.85	100
weighted avg	0.88	0.87	0.87	100

- **مقایسه‌ی شبکه عصبی و ماشین بردار پشتیبان**

در صورت عدم استفاده از minmax scaler دقت مدل svm در حدود ۷۰ درصد خواهد بود. در صورتی که برای ورودی ماشین بردار پشتیبان از نرمال‌سازی minmax استفاده شود، دقت مدل برابر با شبکه عصبی یعنی ۸۷ درصد می‌باشد. بنابراین در این حالت عملکرد این دو مدل تقریبا مشابه است. با این حال شبکه عصبی در تشخیص داده‌های مربوط به بازه‌ی دوم بهتر از مدل SVM عمل کرده است.