MACHINE LEARNING FOUNDATION

CAPSTONE PROJECT REPORT

PREDICTION OF ROOM OCCUPANCY AS A BINARY CLASSIFICATION

Name: Sandun Jayawardhana

Reg No: 241

Contents

1 Dataset

Name: Occupancy Detection Dataset

Source: Luis Candanedo, luismiguel.candanedoibarra '@' umons.ac.be, UMONS.

Link: https://archive.ics.uci.edu/ml/datasets/Occupancy+Detection+

The dataset contains experimental data obtained in one minute frequency to determine occupancy of a room. Data columns include temperature, humidity, light, CO2 and humidity ratio.

Data Set Characteristics:	Multivariate, Time-Series	**Number of Instances:**	20560	**Area:**	Computer
Attribute Characteristics:	Real	**Number of Attributes:**	7	**Date Donated**	2016-02-29
Associated Tasks:	Classification	**Missing Values?**	N/A	**Number of Web Hits:**	182402

The dataset is divided into 03 sets.

01. Training set
02. Testing set 1
03. Testing set 2

The above divided sets can be used for the machine learning model training and testing. Hence, train test split was not done in the project.

1.1 Attributes

- Date time year-month-day hour:minute:second
- Temperature, in Celsius
- Relative Humidity, %
- Light, in Lux
- CO2, in ppm
- Humidity Ratio, Derived quantity from temperature and relative humidity, in kgwater-vapor/kg-air
- Occupancy, 0 or 1, 0 for not occupied, 1 for occupied status

The Occupancy is the target column which need to be predicted using the machine learning algorithm.

1. Methodology

The Google Colab environment was used for the development. Below libraries were used.

1. Pandas
2. Scipy
3. Sklearn
4. Matplotlib
5. Seaborn
6. Numpy

The below steps were followed for building the machine learning model.

1. Loading dataset
2. Data observation
3. Data cleaning
4. Feature selection
5. Model building
6. Model evaluation
7. Model tuning

1.2 Loading dataset

Dataset was loaded directly from web site to the Colab instance using Linux 'wget' command. Then it was unzipped and loaded as a Pandas dataframe.

```
--2022-05-13 18:48:52--  https://archive.ics.uci.edu/ml/machine-learning-databases/00357/occupancy_data.zip
Resolving archive.ics.uci.edu (archive.ics.uci.edu)... 128.195.10.252
Connecting to archive.ics.uci.edu (archive.ics.uci.edu)|128.195.10.252|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 335713 (328K) [application/x-httpd-php]
Saving to: 'occupancy_data.zip'

occupancy_data.zip  100%[===================>] 327.84K  1.54MB/s    in 0.2s

2022-05-13 18:48:52 (1.54 MB/s) - 'occupancy_data.zip' saved [335713/335713]

Archive:  occupancy_data.zip
  inflating: datatest.txt
  inflating: datatest2.txt
  inflating: datatraining.txt
```

1.3 Data observation

The dataset was observed as a sample. Also, it was confirmed that the whole training dataset is loaded successfully. The dimensionality of the dataset was (8143,7).

The loaded dataset columns have below datatypes.

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 8143 entries, 1 to 8143
Data columns (total 7 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   date           8143 non-null   object 
 1   Temperature    8143 non-null   float64
 2   Humidity       8143 non-null   float64
 3   Light          8143 non-null   float64
 4   CO2            8143 non-null   float64
 5   HumidityRatio  8143 non-null   float64
 6   Occupancy      8143 non-null   int64  
dtypes: float64(5), int64(1), object(1)
memory usage: 508.9+ KB
```

1.4 Data cleaning

Data cleaning is a major and an important step of the machine learning model building procedure. Below 02 tasks were for data cleansing.

01. Treating null / Missing values

The selected dataset does not have null or missing values. Therefore, it was not required to handle or drop any data rows in this stage.

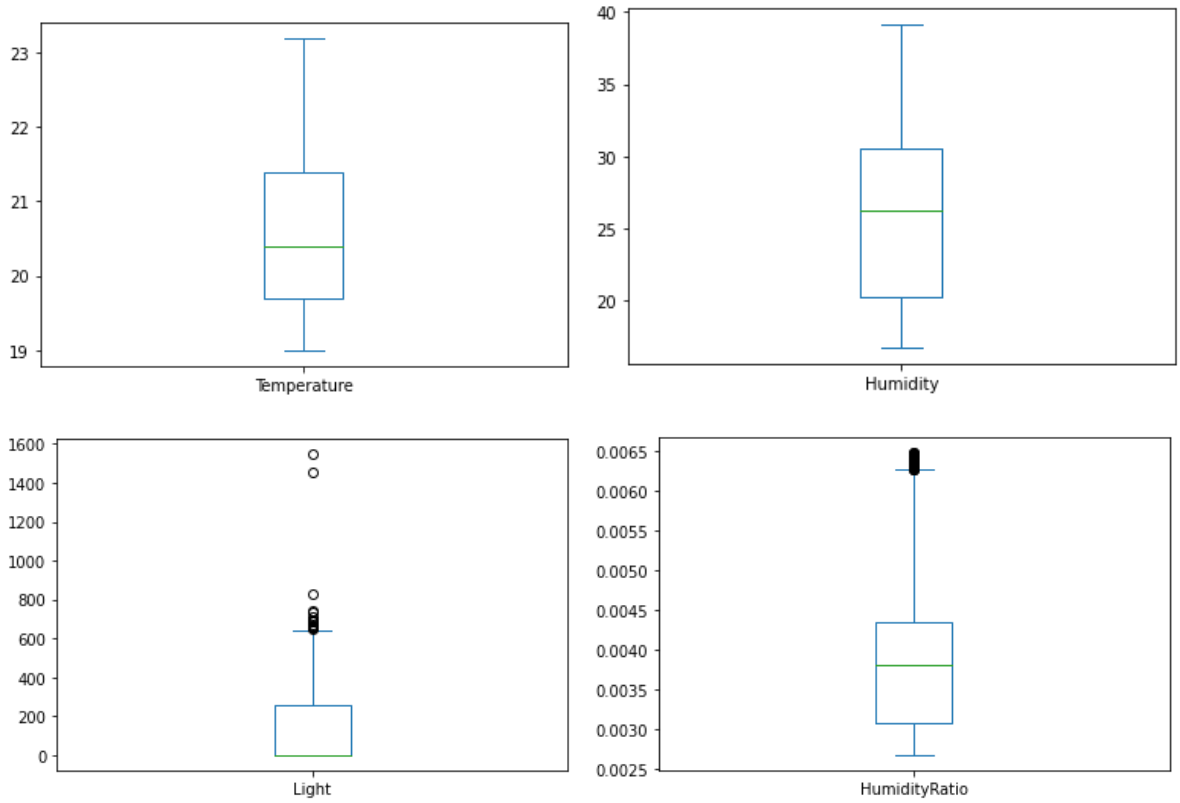
```
data.columns[data.isnull().any()]
```

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

No null values were found in the dataset. Therefore, it is not required to handle null values.

02. Treating outliers

Box plots were used to identify outliers of all the feature columns. Few bar plots are shown below.



It can be observed that ‘Light’ and ‘Humidity Ratio’ has outliers. Those outliers were filtered using quantile-based filtering.

```
for column in ['Light', 'HumidityRatio']:
  Q1 = data[column].quantile(0.25)
  Q3 = data[column].quantile(0.75)
  IQR = Q3 - Q1

  data=data[(data[column]>Q1-1.5*IQR) & (data[column]<Q3+1.5*IQR)]
  plt.figure()
  data[column].plot(kind='box')
  plt.show()
```

1.5 Balancing the dataset

The dataset was observed for imbalance as the next step. It is observed that the value counts of target column are imbalanced as follows.

```
[ ]  data['Occupancy'].value_counts()

     0    6410
     1    1597
     Name: Occupancy, dtype: int64
```

The dataset was balanced using down sampling of value 0 counting which is the majority class.

```
data = data.drop(data[data['Occupancy'] == 0].sample(frac=.6).index)
data['Occupancy'].value_counts()

0    2564
1    1597
Name: Occupancy, dtype: int64
```

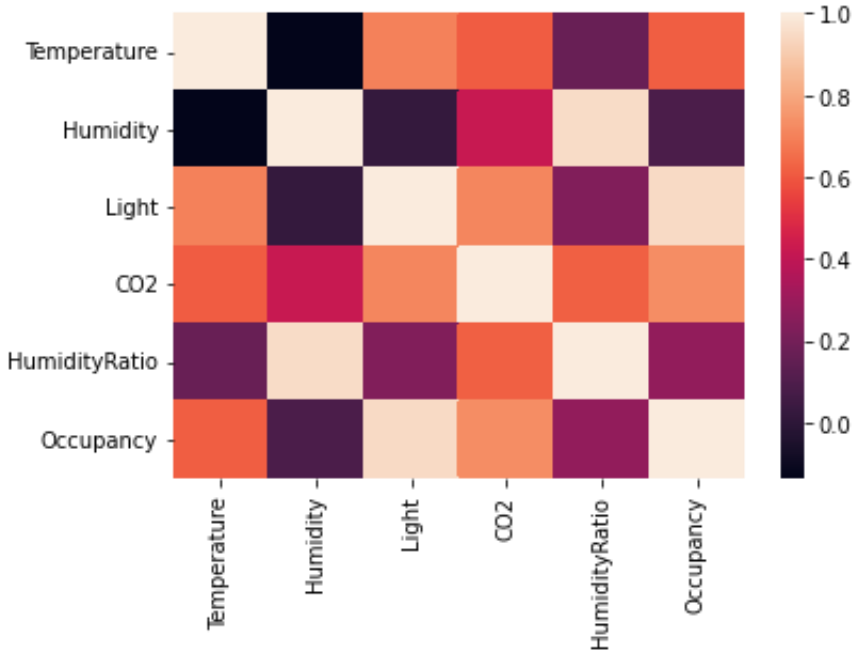
1.6 Feature selection

There are 07 columns in the dataset. The target column is set as 'Occupancy'. Out of the remaining 06 columns, index column is the datetime which was not considered as a feature. Therefore, 05 columns are available for feature selection.

As the first step, dataset was described using data.describe() function.

	Temperature	Humidity	Light	CO2	HumidityRatio	Occupancy
count	4161.000000	4161.000000	4161.000000	4161.000000	4161.000000	4161.000000
mean	20.828097	25.664226	193.172711	673.210180	0.003903	0.383802
std	1.035118	5.394723	221.999744	317.311131	0.000837	0.486369
min	19.000000	16.790000	0.000000	412.750000	0.002682	0.000000
25%	20.000000	20.545000	0.000000	441.500000	0.003191	0.000000
50%	20.790000	26.200000	7.000000	471.000000	0.003811	0.000000
75%	21.700000	30.200000	444.000000	854.000000	0.004445	1.000000
max	23.180000	38.145000	638.000000	1879.250000	0.006257	1.000000

It is observed that all the columns have acceptable variance for considering them to the next step. As the next step, correlation matrix was observed.



According to the correlation matrix, it can be observed that 'Temperature', 'Light' and 'CO2' columns have higher correlation with 'Occupancy' column. 'Humidity' and 'Humidity Ratio' have lower correlation values.

However, it was decided to keep all the columns as features because of the low number of column count. The backward elimination will be used in the model tuning step.

1.7 Model building

Since this is classification problem, 03 classification algorithms are used.

01. Random Forrest classifier
02. Decision Tree Classifier
03. K-Nearest Neighbour classifier

All the models were initiated and fit with training data as follows.

Random Forrest Classifier

```
[ ]  from sklearn.ensemble import RandomForestClassifier

     X_train = data[X_variables]
     y_train = data[y_variable].values

     RFClassifier1 = RandomForestClassifier().fit(X_train, y_train)
```

Decision tree classifier

```
[ ]  from sklearn.tree import DecisionTreeClassifier

     DTClassifier1 = DecisionTreeClassifier().fit(X_train,y_train)
```

K-Nearest Neighbour Classifier

```
[ ]  from sklearn.neighbors import KNeighborsClassifier

     KNNClassifier1 = KNeighborsClassifier(n_neighbors=2).fit(X_train,y_train)
```

1.8 Model evaluation

The dataset included a test dataset itself. It was loaded and features were extracted to be fed into the model.

```
data_test = pd.read_csv("datatest.txt")
print(data_test.shape)
```

```
(2665, 7)
```

The built models were evaluated against the test data set using Sklearn metrics function. The results are shown below.

Model	Area under ROC	Precision		Recall		f1-score	
		Class '0'	Class '1'	Class '0'	Class '1'	Class '0'	Class '1'
Random Forrest Classifier - 05 features	0.94	0.95	0.94	0.97	0.91	0.96	0.93
Decision Tree Classifier - 05 features	0.85	0.86	0.94	0.97	0.74	0.92	0.82
K-Nearest Neighbour Classifier - 05 features	0.93	0.94	0.95	0.97	0.9	0.96	0.9

```
roc_auc = metrics.roc_auc_score(y_test, y_pred_rfc)
print(roc_auc)

results = metrics.classification_report(y_test, y_pred_rfc)
print(results)
```

```
0.9388607531860798
              precision    recall  f1-score   support

           0       0.95      0.97      0.96      1693
           1       0.94      0.91      0.93       972

    accuracy                           0.95      2665
   macro avg       0.95      0.94      0.94      2665
weighted avg       0.95      0.95      0.95      2665
```

1.9 Model tuning

The model was tuned by alternating the features set. Backward elimination was used for this, and 'Humidity' column was removed from the feature set.

```
X_variables = ['Temperature', 'Light', 'CO2', 'HumidityRatio']
y_variable = 'Occupancy'

print(F"X_variables = {X_variables}")
print(F"y_variable = {y_variable}")

X_train = data[X_variables]
y_train = data[y_variable].values

RFClassifier2 = RandomForestClassifier().fit(X_train, y_train)
DTClassifier2 = DecisionTreeClassifier().fit(X_train,y_train)
KNNClassifier2 = KNeighborsClassifier(n_neighbors=2).fit(X_train,y_train)
```

Model	Area under ROC	Precision		Recall		f1-score	
		Class '0'	Class '1'	Class '0'	Class '1'	Class '0'	Class '1'
Random Forrest Classifier - 04 features	0.94	0.95	0.94	0.97	0.92	0.96	0.93
Decision Tree Classifier -04 features	0.89	0.9	0.96	0.98	0.82	0.94	0.88
K-Nearest Neighbor Classifier - 04 features	0.93	0.94	0.94	0.93	0.94	0.94	0.94

1.10 Model saving

The best model was saved for deployment using pickle. This model can be deployed standalone and exposed as via an API.

2 Discussion

The machine learning problem which I have been trying to solve is to predict the room occupancy using the temperature, light, CO2 percentage and the humidity ratio. The model evaluation results show acceptable and good performance.

First, we will consider the raw models which were not tuned. The ROC values of all the models are higher than 0.85 which is a good indication. It is observed that all models have higher precision and recall values over both target variable classes. Hence, it can be assumed that the data preprocessing steps are properly done.

The Random Forrest classifier has outperformed other 02 models in terms of precision, recall and f1-score. Therefore, we can state that the model's false positives and true negatives are lower. Also, the model is balanced with both majority and minority classes of the target variable having similar metric values. The dataset balancing should have caused this balanced performance.

Then we will compare the tuned models' performance with initial models. It can be observed that the model metrics have slight improvements. The initial models had acceptable higher metrics therefore this is expected. We cannot expect drastic model improvements when the initial models already show good performance. The best model is selected as tuned Random Forrest Classifier. It can be used to accurately predict the room occupancy when temperature, light, CO2 percentage and humidity ratio are input.

Further improvements may be possible with new features derived from the present features. Also, data normalization can be done before feeding to the models.

3 Conclusion

The machine learning model building pipeline was well followed during the project. It included dataset preparation, data preprocessing, model building and evaluation, model tuning and presenting the results. Each step has its importance to provide a better solution ultimately.

The results show that model can be deployed and used in production to give better predictions. It predicts both classes accurately. Hence, I think the outcome of the project is delivered.