MBA 652A STATISTICAL MODELLING FOR BUSINESS ANALYTICS

PROJECT REPORT ON

ROOM OCCUPANCY DETECTION

SUBMITTED TO:

Dr. DEVLINA CHATTERJEE

SUBMITTED BY:

Mr. FARAZ AHMAD KHAN (19114006)

Mr. YASH KALPESH PANCHAL (19114020)

Mr. ASHWIN BHIDE (19125015)

Ms. ANUSHRI SINGH (19125010)

OBJECTIVE:

The objective of the project is to determine whether a given room is occupied or not, meaning whether there is some person present in the room or not based on various variables. We will be using non-linear models such as Logit and Probit in detecting the Room Occupancy.

METHODOLOGY:

We have used Logit Regression model and Probit Regression Model to predict the Binary Dependent Variable – Occupancy. Also, we have used Linear Probability Model to classify the dependent variable to compare linear and non-linear model.

DATA DESCRIPTION:

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

Dataset: Occupancy Detection Data Set

Attribute Information:

Independent variables

- **Date** – date and time of the readings taken for a given room
- **Temperature** – temperature of the room in degree Celsius
- **Humidity** – relative humidity of the room
- **Light** – intensity of light in lux
- **CO2** – carbon dioxide level in the room in ppm
- **HumidityRatio** - derived quantity from temperature and relative humidity, in kgwater-vapor/kg-air

Dependent variables

- **Occupancy** – binary variable (0 if room is not occupied, 1 if occupied)

Scatter Plots:

Please refer to Annexure I for all scatter plots

Glimpse of Data:

	date	Temperature	Humidity	Light	CO2	HumidityRatio	Occupancy
0	2/2/2015 14:19	23.7000	26.272	585.200000	749.200000	0.004764	1
1	2/2/2015 14:19	23.7180	26.290	578.400000	760.400000	0.004773	1
2	2/2/2015 14:21	23.7300	26.230	572.666667	769.666667	0.004765	1
3	2/2/2015 14:22	23.7225	26.125	493.750000	774.750000	0.004744	1
4	2/2/2015 14:23	23.7540	26.200	488.600000	779.000000	0.004767	1

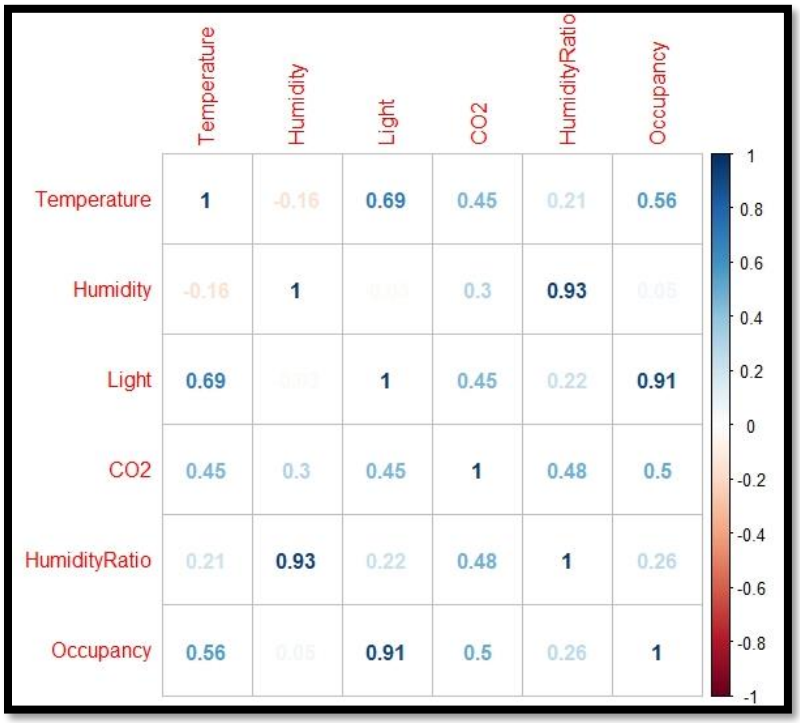
Here, date is just a non-informative variable we could just exclude it for over analysis.

Summary of Data:

	Temperature	Humidity	Light	CO2	HumidityRatio	Occupancy
count	20560.000000	20560.000000	20560.000000	20560.000000	20560.000000	20560.000000
mean	20.906212	27.655925	130.756622	690.553276	0.004228	0.231031
std	1.055315	4.982154	210.430875	311.201281	0.000768	0.421503
min	19.000000	16.745000	0.000000	412.750000	0.002674	0.000000
25%	20.200000	24.500000	0.000000	460.000000	0.003719	0.000000
50%	20.700000	27.290000	0.000000	565.416667	0.004292	0.000000
75%	21.525000	31.290000	301.000000	804.666667	0.004832	0.000000
max	24.408333	39.500000	1697.250000	2076.500000	0.006476	1.000000

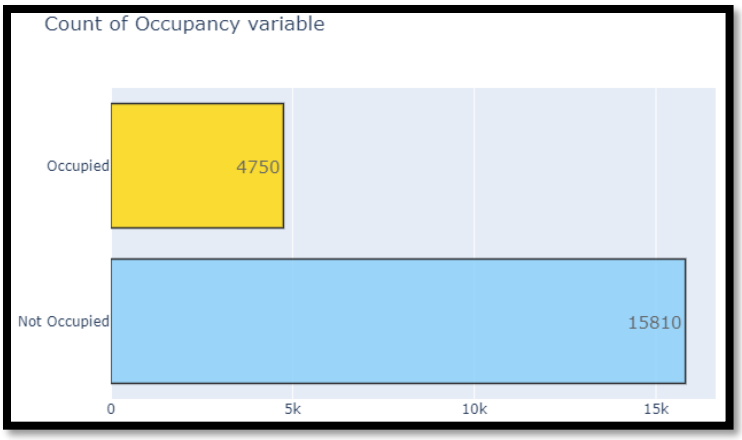
CORRELATION MATRIX:

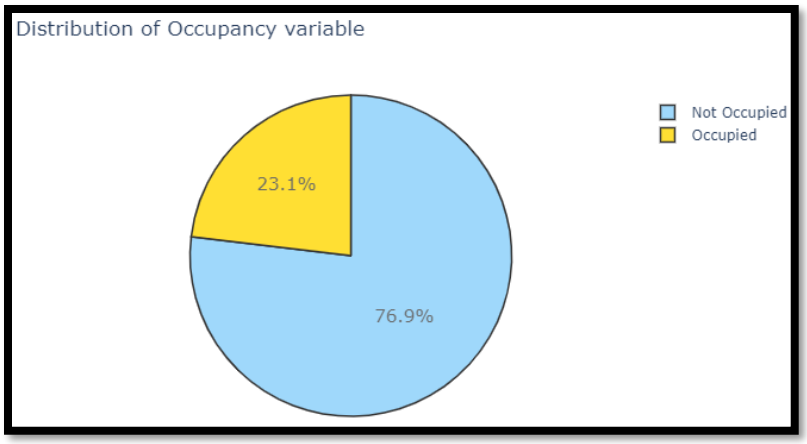
Below graph depicts correlation of each variable with other including Binary Dependent Variable.



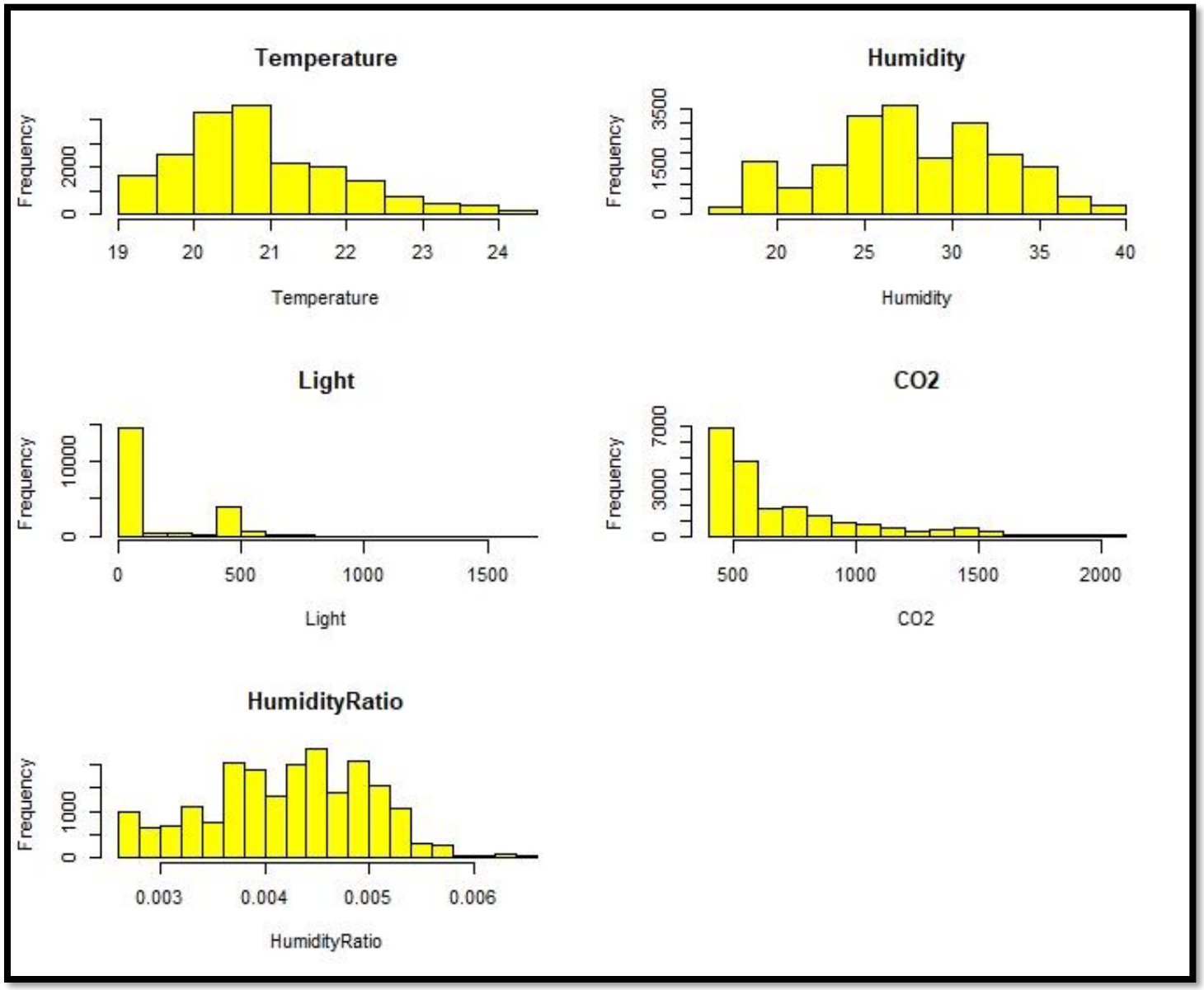
Visualization of Data:

Count and Distribution of Binary Dependent Variable (Occupancy) is shown in below 2 graphs,





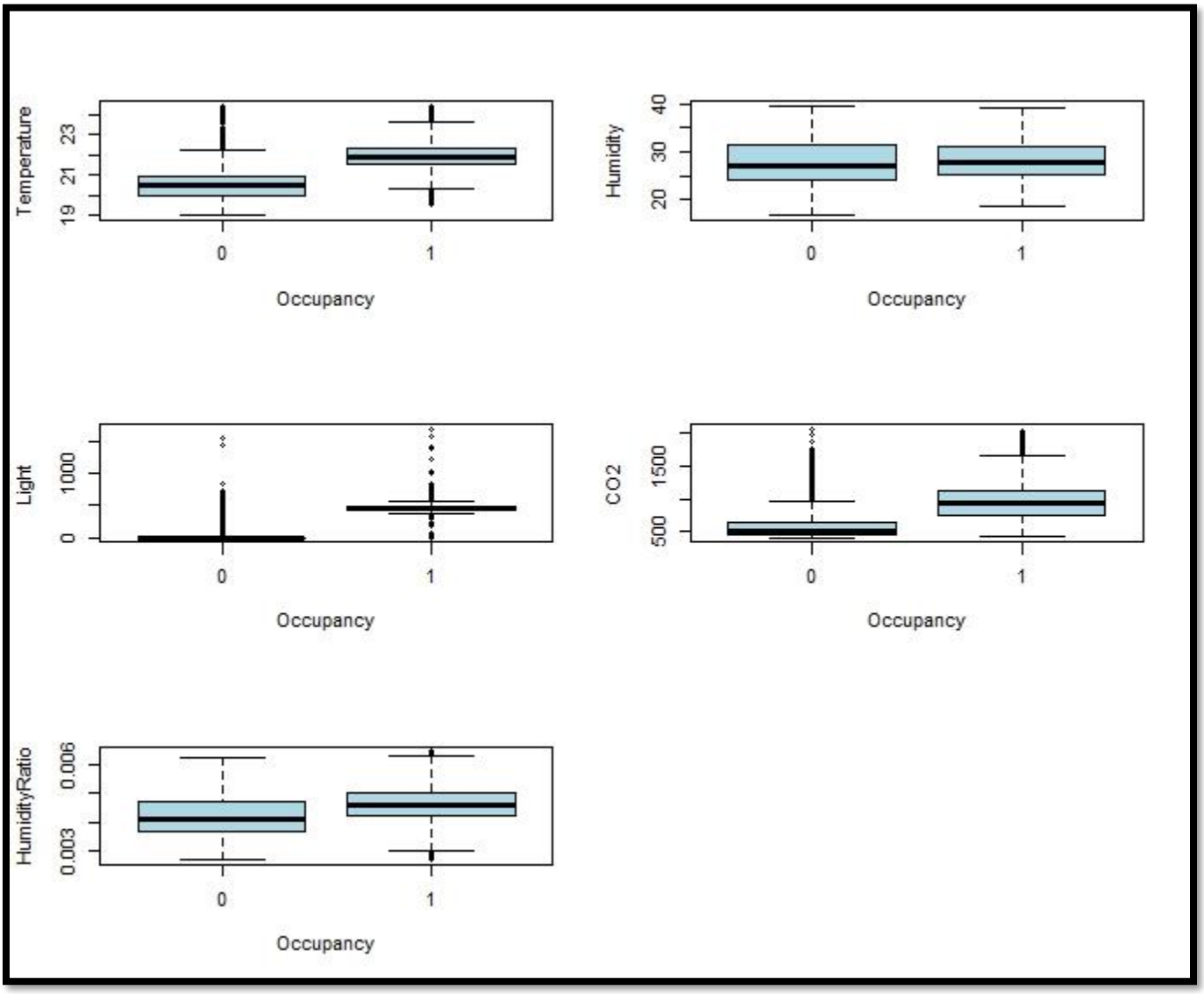
Distribution of independent variables is shown below,



It is evident from above plot that CO2 content is positive skewed, and also around 50% of the rooms are having light with 0 lux saying that those rooms are having lights switched off and it is night when the data is collected.

While, other variables are more or less normally distributed.

Boxplot of independent variables,



It can be evaluated from the boxplots that, occupied rooms have higher mean temperature than those which are not occupied, same follows for CO2, Humidity, HumidityRatio. Also, is can seen that if a person is present in a room than it has more chances of switching on the light which is seen in the boxplot of Light.

Model Application:

After, describing the data and getting an idea how the variables behave we will build non-linear models to predict the dependent binary variable.

We have made a division of data points into train and test in ratio of 1:1, i.e., we have 10280 data points in training set and another 10280 data points in testing set, split is done randomly

Probit Regression Model:

It uses Cumulative Probability Function (CDF) for determine the outcome of dependent Variable, it can be mathematically represented as,

$$\Pr(Y = 1|X) = \emptyset(Z)$$

$$Z = \beta_0 + \beta_1 * X_1 + \beta_2 * X_2 + \cdots$$

Y = Binary Dependent Variable,

X_i = Independent Variables,

β_i = coefficients

Φ = cumulative probability functions,

Based on our data set we, best fit Probit Regression Model is,

$$\Pr(Occupancy) = \emptyset(4.0622 - 0.4871 * Temperature + 0.0407 * Humidity + 0.0122 * Light + 0.0018 * CO2)$$

Summary and Observations:

Below, mentioned image depicts the summary of above model,

Probit Regression Results

Dep. Variable:	Occupancy	**No. Observations:**	10280
Model:	Probit	**Df Residuals:**	10275
Method:	MLE	**Df Model:**	4
Date:	Thu, 11 Jun 2020	**Pseudo R-squ.:**	0.8942
Time:	17:38:44	**Log-Likelihood:**	-585.06
converged:	True	**LL-Null:**	-5532.4
Covariance Type:	nonrobust	**LLR p-value:**	0.000

	coef	std err	z	P>|z|	[0.025	0.975]
const	4.0622	1.030	3.942	0.000	2.043	6.082
Temperature	-0.4871	0.047	-10.262	0.000	-0.580	-0.394
Humidity	0.0407	0.011	3.849	0.000	0.020	0.061
Light	0.0122	0.000	40.840	0.000	0.012	0.013
CO2	0.0018	0.000	10.463	0.000	0.001	0.002

- HumidityRatio is not present in the model equation as there is a high correlation between Humidity and HumidityRatio, so either of them one is sufficient to explain the Binary Dependent Variable.
- Coefficient of CO2 is very small compared to others as the value of CO2 in ppm is in range of 500-1000, which is large compared to other variables such as Temperature which is in range of 15-30.
- Maximum Likelihood Estimator method is used to predict the coefficients.
- Log-Likelihood = -585.06, maximized value of the log-likelihood function
- LL-Null = -5532.4, maximized log-likelihood function when only an intercept is included.
- Pseudo R-squared = 0.8942, meaning that the model is very good fit as it is close to 1, the independent variables are good enough to explain the dependent variable.

Logit Regression Model:

This model is popularly known as Logistic Regression Model, which uses sigmoid function to predict the probability of Binary Dependent Variable, mathematically it can be represented as,

$$\Pr(Y = 1|X) = \frac{1}{1 + e^{-z}}$$

$$Z = \beta_0 + \beta_1 * X_1 + \beta_2 * X_2 + \cdots$$

Y = Binary Dependent Variable,

X_i = Independent Variables,

β_i = coefficients

Φ = cumulative probability functions,

Best fit Logit model for the given data set is,

$$\Pr(Occupancy) = \frac{1}{1 + e^{-(4.3288 - 0.7863 * Temperature + 0.0762 * Humidity + 0.0236 * Light + 0.0036 * CO2)}}$$

Summary and Observations:

Summary of the best fit Logit Regression Model is as follows,

Logit Regression Results

Dep. Variable:	Occupancy	**No. Observations:**	10280
Model:	Logit	**Df Residuals:**	10275
Method:	MLE	**Df Model:**	4
Date:	Thu, 11 Jun 2020	**Pseudo R-squ.:**	0.9034
Time:	17:47:15	**Log-Likelihood:**	-534.19
converged:	True	**LL-Null:**	-5532.4
Covariance Type:	nonrobust	**LLR p-value:**	0.000

	coef	std err	z	P>|z|	[0.025	0.975]
const	4.3288	2.150	2.013	0.044	0.115	8.543
Temperature	-0.7863	0.101	-7.766	0.000	-0.985	-0.588
Humidity	0.0762	0.022	3.537	0.000	0.034	0.118
Light	0.0236	0.001	30.663	0.000	0.022	0.025
CO2	0.0036	0.000	9.599	0.000	0.003	0.004

- Observations are very much similar to that of best fir Probit Regression Model
- Maximum Likelihood Estimator method is used to predict the coefficients.
- Log-Likelihood = -534.19, maximized value of the log-likelihood function
- LL-Null = -5532.4, maximized log-likelihood function when only an intercept is included.
- Pseudo R-squared = 0.9034, meaning that the model is very good fir as it is close to 1, the independent variables are good enough to explain the dependent variable.
- This is because both the models are quite efficient non-linear models for explaining the Binary Dependent Variable.

Test and Results:

Variation Inflation Factor:

As both the results are having the same variables in the final model, we would perform VIF test on these variables.

Results of VIF test is given in the table below,

Attribute	Temperature	Light	CO2	Humidity
VIF	2.17	1.97	1.59	1.24

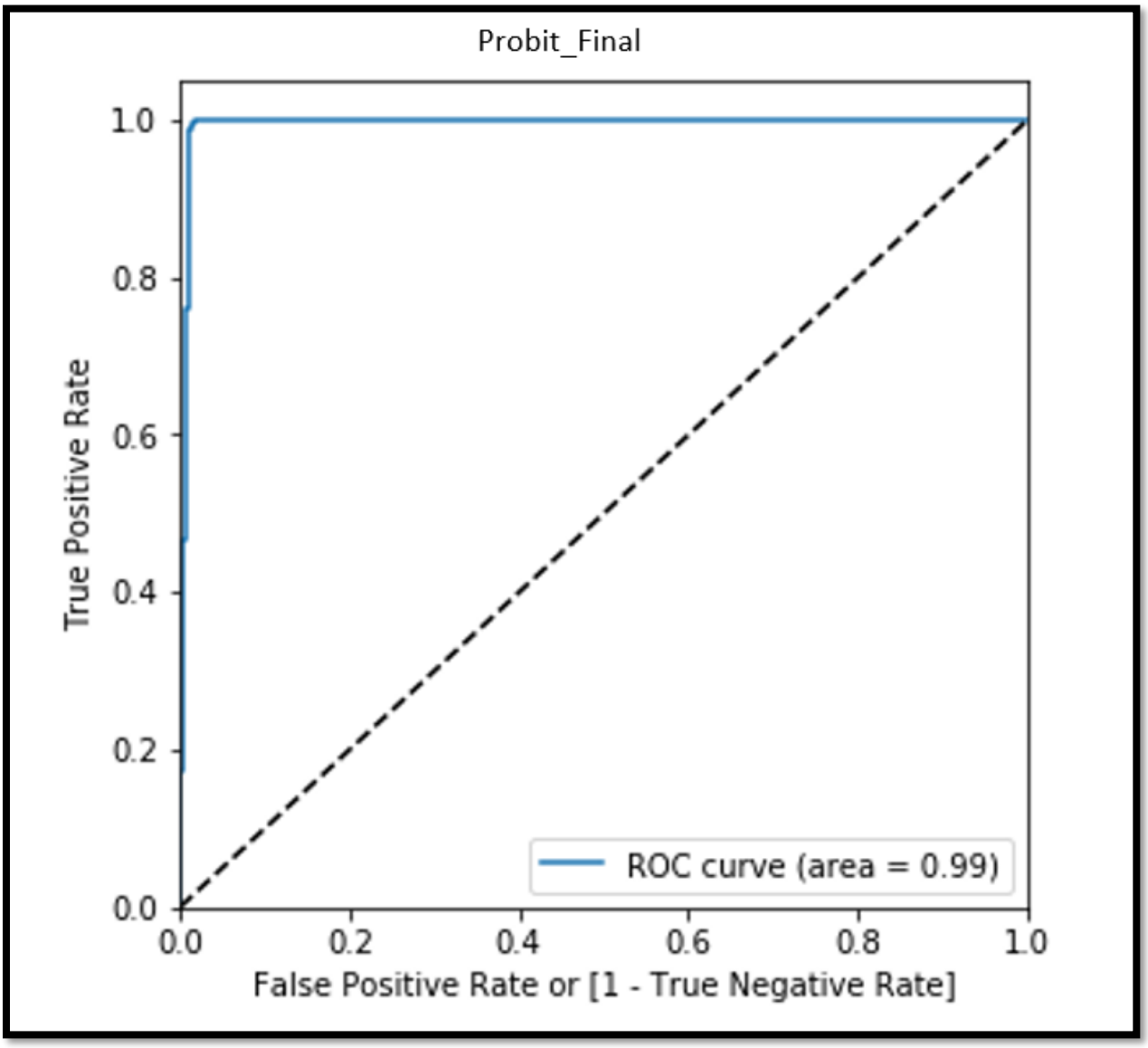
Variable Significance Test:

As all the variables in the final model of Probit Regression and Logit Regression are having p-value < 0.05, we can conclude that all the variables in these models are statistically significant at 95% level.

ROC Curve:

Probit Model:

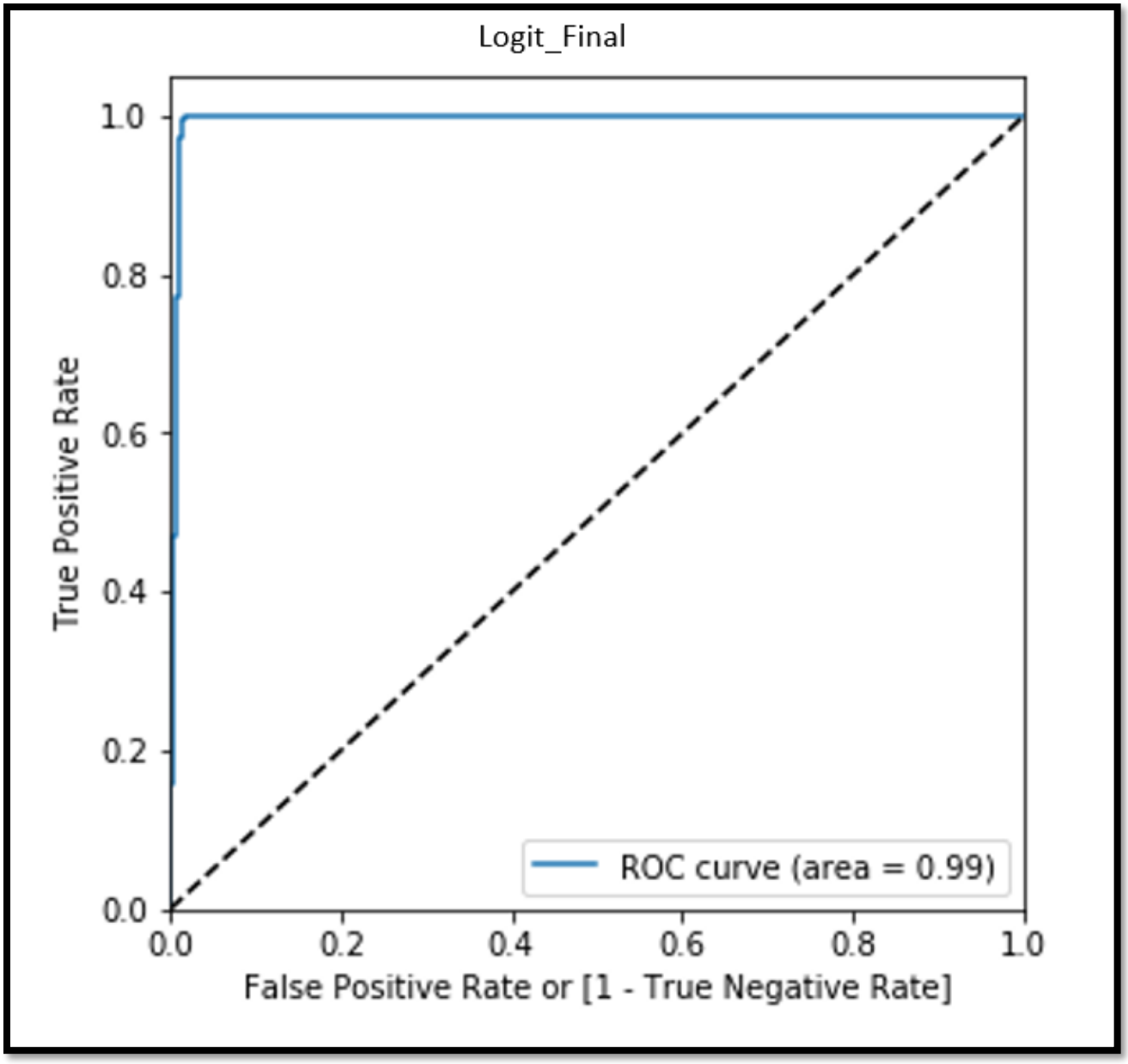
Below image shows the ROC curve for best fit Probit Model,



Area Under the Curve (AUC) = 0.99

Logit Model:

ROC curve for Logit Regression best fit model,



Area Under the Curve (AUC) = 0.99

Classification Table:

Probit Model:

Confusion Matrix is formed as shown in table below, if probability of Y(Occupancy) is greater then or equal to cutoff than it is predicted as positive and if it is less than cutoff it is predicted as negative.

Taking cutoff = 0.5

		Predicted	
		Neg (0)	Pos (1)
Observed	Neg (0)	7791	98
	Pos (1)	10	2381

Therefore, from the table we can calculate

Precision	0.997
Recall	0.955
F1 score	0.976
Accuracy	0.988
False Positive Rate	0.014
True Positive Rate	0.997
Test Specificity	0.986
Test Sensitivity	0.997

Logit Model:

Confusion Matrix is formed as shown in table below, if probability of Y(Occupancy) is greater then or equal to cutoff than it is predicted as positive and if it is less than cutoff it is predicted as negative.

Taking cutoff = 0.5

		Predicted	
		Neg (0)	Pos (1)
Observed	Neg (0)	7791	98
	Pos (1)	11	2380

Therefore, from the table we can calculate

Precision	0.997
Recall	0.955
F1 score	0.976
Accuracy	0.988
False Positive Rate	0.014
True Positive Rate	0.997
Test Specificity	0.986
Test Sensitivity	0.997

Conclusion:

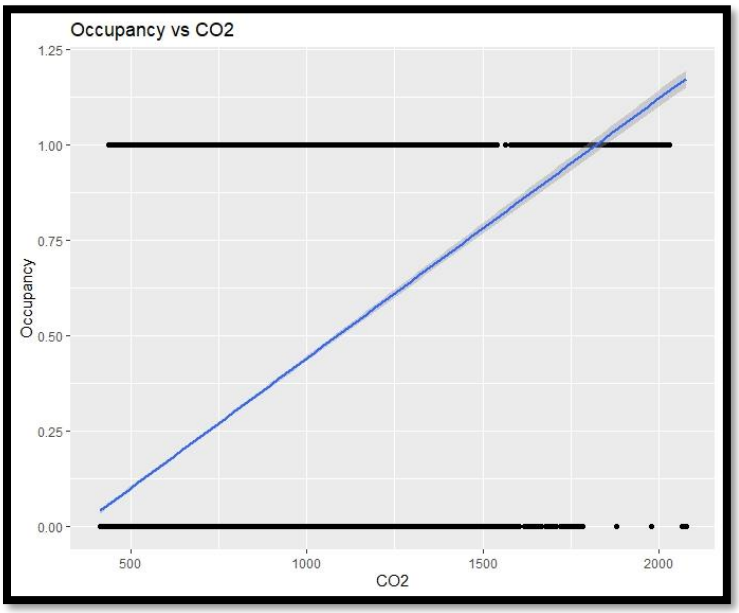
- Both the models, Logit Regression and Probit Regression are behaving in exactly the same manner, which is rare but at least it is expected that both models behave in same manner, but the level of similarity is due to the data set.
- Accuracy achieved by both models is same which is 0.988.
- Pseudo R square for both models is quite high (close to 1) that means the models are very good fit and they are explaining the Binary Dependent Variable efficiently.
- For Logit Regression Model Pseudo R square (0.9034) is slightly higher than that of Probit Model (0. 8942), therefore we can say that Logit model has a slight edge over Probit for this particular data set.
- Humidity Ratio is not included in the both the final models, as there exists a high multi collinearity between Humidity and Humidity Ratio, so any one of 2 is good enough the capture the essence in explaining the Binary Dependent Variable.
- ROC curves of both models are same as we can expect that.
- We can use any one of two non-linear models to predict Binary Dependent Variable as both are given almost equivalent result for other data sets.

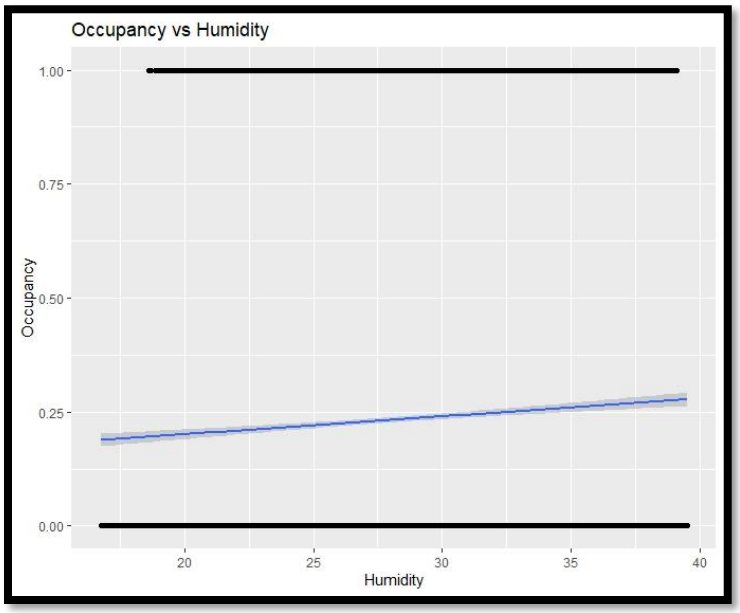
Reference:

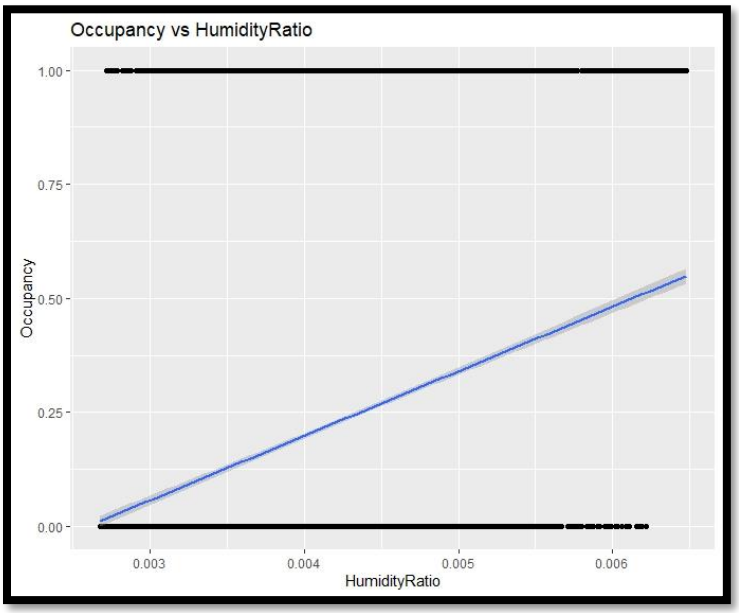
- https://archive.ics.uci.edu/ml/datasets/Occupancy+Detection+

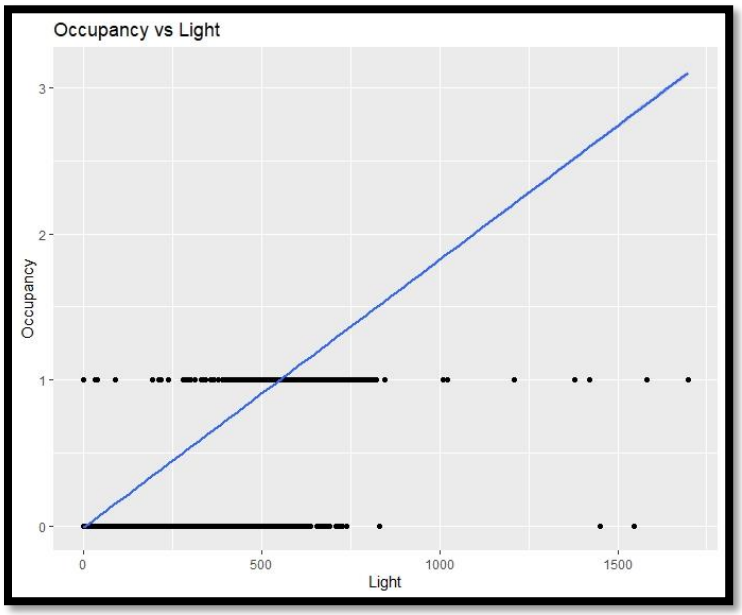
- https://en.wikipedia.org/wiki/Probit_model#:~:text=In%20statistics%2C%20a%20probit%20model,%2C%20coming%20from%20probability%20%2B%20unit.
- https://en.wikipedia.org/wiki/Logistic_regression
- https://www.youtube.com/watch?v=n9ukDRQ1mCI&t=3120s
- https://www.statsmodels.org/stable/index.html
- https://www.geeksforgeeks.org/
- http://www.sthda.com/english/articles/39-regressionmodel-diagnostics/160-multicollinearity-essentials-and-vif-in-r/

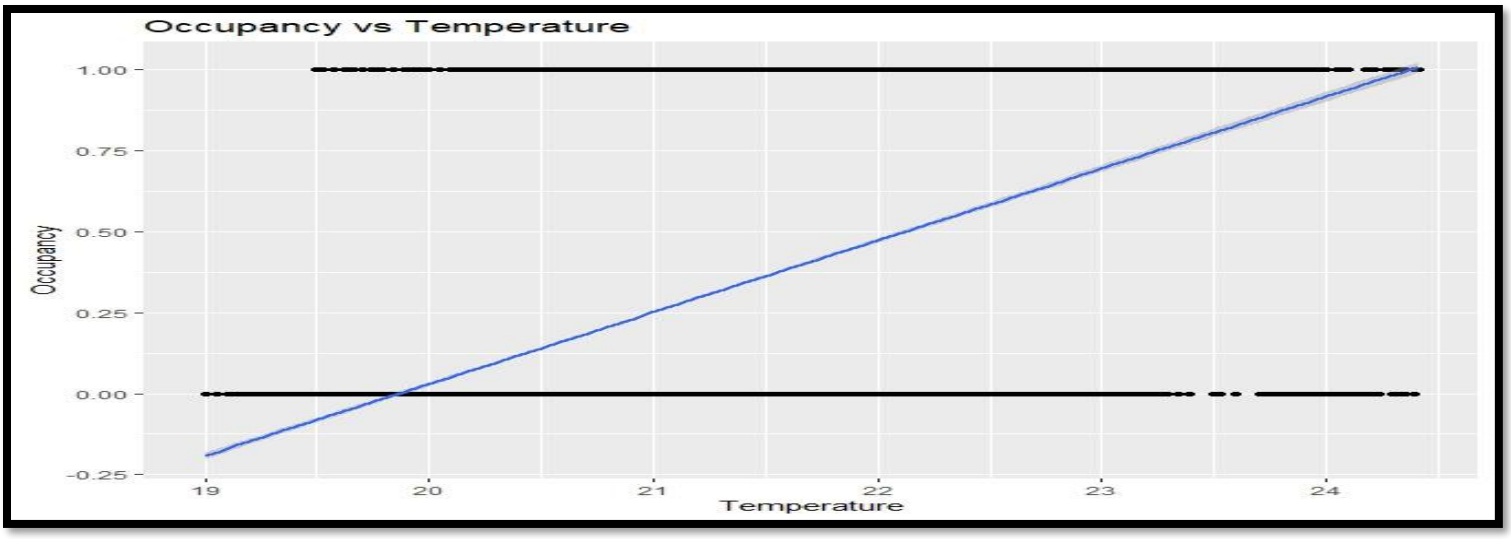
Annexure 1:











Annexure 2:

Linear Probability Model (LPM) was also applied on the given data set and it also yielded almost same results,

Best fit LPM is,

$$\Pr(Occupancy) = 1.2830 - 0.0688 * Temperature + 0.002 * Light + 0.0002 * CO2$$

Summary of the above mentioned LPM is,

OLS Regression Results

Dep. Variable:	Occupancy	**R-squared:**	0.866
Model:	OLS	**Adj. R-squared:**	0.866
Method:	Least Squares	**F-statistic:**	2.220e+04
Date:	Fri, 12 Jun 2020	**Prob (F-statistic):**	0.00
Time:	00:59:12	**Log-Likelihood:**	4668.3
No. Observations:	10280	**AIC:**	-9329.
Df Residuals:	10276	**BIC:**	-9300.
Df Model:	3		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	1.2830	0.041	31.527	0.000	1.203	1.363
Temperature	-0.0688	0.002	-34.086	0.000	-0.073	-0.065
Light	0.0020	1.02e-05	191.560	0.000	0.002	0.002
CO2	0.0002	5.67e-06	34.410	0.000	0.000	0.000

Omnibus:	8883.558	**Durbin-Watson:**	1.999
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	722883.582
Skew:	-3.730	**Prob(JB):**	0.00
Kurtosis:	43.398	**Cond. No.**	2.07e+04

Observe that here, only 3 significant variables are present in the final model, Humidity and HumidityRatio are both excluded in determining the probability of Binary Dependent Variable, this is because Temperature and Humidity are somewhat related, not highly

but related, therefore Temperature Variable captures the essence of explaining the Binary Dependent Variable and is sufficient enough, but this was not in the case of Logit and Probit Model.

Annexure 3:

Python Code:

```
#!/usr/bin/env python
# coding: utf-8

# # Importing necessary libraries

# In[217]:


import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.api as sm
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
import scikitplot as skplt
from sklearn import metrics
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score
from sklearn.feature_selection import RFE
from sklearn.linear_model import LinearRegression
from statsmodels.stats.outliers_influence import variance_inflation_factor
from sklearn.metrics import classification_report, accuracy_score, auc
from sklearn import preprocessing
import plotly.graph_objs as go
import plotly.offline as py
get_ipython().run_line_magic('matplotlib.inline', '')


# # Reading file

# In[134]:


path = "E:\\IIT Kanpur\\IME 2nd Sem Courses\\MBA652-SMBA\\Projects\\Binary Dependent Variable\\Occupancy of Room\\datatest.csv"

dataset = pd.read_csv(path)
dataset.head()


# In[135]:


dataset = dataset.drop(['date'], axis = 1)


# In[136]:


dataset.head()
```

Correlation Matrix

In[137]:

```
dataset.corr()
```

Plots

In[218]:

```
D = dataset[(dataset["Occupancy"] != 0)]
H = dataset[(dataset["Occupancy"] == 0)]
```

In[219]:

```
#------------COUNT-----------------------
def target_count():
    trace = go.Bar( x = dataset['Occupancy'].value_counts().values.tolist(),
                    y = ['Not Occupied','Occupied' ],
                    orientation = 'h',
                    text = dataset['Occupancy'].value_counts().values.tolist(),
                    textfont = dict(size = 15),
                    textposition = 'auto',
                    opacity = 0.8,marker = dict(
                    color = ['lightskyblue', 'gold'],
                    line = dict(color = '#000000',width=1.5)))

    layout = dict(title =  'Count of Occupancy variable')

    fig = dict(data = [trace], layout=layout)
    py.iplot(fig)

#------------PERCENTAGE--------------------
def target_percent():
    trace = go.Pie(labels = ['Not Occupied','Occupied'], values = dataset['Occupancy'].value_counts(),
                   textfont = dict(size = 15), opacity = 0.8,
                   marker = dict(colors = ['lightskyblue', 'gold'],
                                 line = dict(color = '#000000', width = 1.5)))


    layout = dict(title =  'Distribution of Occupancy variable')

    fig = dict(data = [trace], layout=layout)
    py.iplot(fig)
```

In[220]:

```
target_count()
target_percent()


# # Train Test Split

# In[138]:


X = dataset.iloc[:, :5]
Y = dataset.iloc[:, 5]


# In[139]:


X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.5, random_state=1)


# In[140]:


X_train = X_train.reset_index(drop=True)
X_test = X_test.reset_index(drop=True)
Y_train = Y_train.reset_index(drop=True)
Y_test = Y_test.reset_index(drop=True)


# # Feature Scaling Training Data
#
# ##  (X-min) / (max - min)

# In[141]:


x = X_train.values #returns a numpy array
min_max_scaler = preprocessing.MinMaxScaler()
x_scaled = min_max_scaler.fit_transform(x)
X_train_scale = pd.DataFrame(x_scaled)


# In[142]:


col_names = ['Temperature', 'Humidity', 'Light', 'CO2', 'HumidityRatio']
X_train_scale.columns = col_names


# In[143]:


X_train_scale = sm.add_constant(X_train_scale)


# # Linear Probability Model
```

Model Application

In[145]:

```
X_LPM = X_train_scale.copy()
```

In[146]:

```
LPM_1 = sm.OLS(Y_train, X_LPM)
result_LPM1 = LPM_1.fit()
result_LPM1.summary()
```

Observation : All variables are significant

Variable Inflation Factor (VIF) calculation

In[147]:

```
def calculate_vif(df):
    vif = pd.DataFrame()
    vif['Features'] = df.columns
    vif['vif'] = [variance_inflation_factor(df.values, i) for i in range(df.shape[1])]
    vif['vif'] = round(vif['vif'],2)
    vif = vif.sort_values(by='vif', ascending=False)
    print(vif)
```

In[148]:

```
calculate_vif(X_LPM)
```

Observation :: HumidityRatio is having high multicollinearity, so dropping it.

In[149]:

```
X_LPM = X_LPM.drop(['HumidityRatio'], axis = 1)
```

Building another model after dropping HumidityRatio

In[150]:

```
LPM_2 = sm.OLS(Y_train, X_LPM)
result_LPM2 = LPM_2.fit()
result_LPM2.summary()
```

Observation : Humidity has p-value > 0.05, therefore it is insignificant, dropping it

In[151]:

```
X_LPM = X_LPM.drop(['Humidity'], axis = 1)
```

Building another model after dropping Humidity

In[152]:

```
LPM_3 = sm.OLS(Y_train, X_LPM)
result_LPM3 = LPM_3.fit()
result_LPM3.summary()
```

Observation : All variables are significant

Variable Inflation Factor (VIF) calculation

In[153]:

```
calculate_vif(X_LPM)
```

High multicollinearity between variables is absent

Final Linear Probability Model

In[154]:

```
X_LPM_colnames = ['Temperature', 'Light', 'CO2']
```

In[155]:

```
X_train_LPM = X_train.copy()
X_train_LPM = X_train_LPM[X_LPM_colnames]
X_train_LPM = sm.add_constant(X_train_LPM)
```

In[158]:

```
LPM_final = sm.OLS(Y_train, X_train_LPM)
result_LPM_final = LPM_final.fit()
result_LPM_final.summary()
```

Best fit Linear Probability model is,

Y = 1.2830 - 0.0688 * Temperature + 0.002 * Light + 0.0002 * CO2

Evalution of Results

1. ROC Curve

In[159]:

```
X_test_LPM = X_test.copy()
X_test_LPM = X_test_LPM[X_LPM_colnames]
X_test_LPM = sm.add_constant(X_test_LPM)
```

In[160]:

```
def draw_roc_curve(actual, probs):
    fpr, tpr, thresholds = roc_curve(actual, probs)
    auc_score = metrics.roc_auc_score( actual, probs )
    plt.figure(figsize=(5, 5))
    plt.plot( fpr, tpr, label='ROC curve (area = %0.2f)' % auc_score )
    plt.plot([0, 1], [0, 1], 'k--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate or [1 - True Negative Rate]')
    plt.ylabel('True Positive Rate')
    plt.legend(loc="lower right")
    plt.show()
```

In[161]:

```
Y_LPM_probs = result_LPM_final.predict(X_test_LPM)
Y_LPM_probs = Y_LPM_probs.to_numpy()
Y_actual = Y_test.to_numpy()
```

In[162]:

```
draw_roc_curve(Y_actual, Y_LPM_probs)
```

2. Confusion Matrix

In[163]:

```
Y_pred_LPM = result_LPM_final.predict(X_test_LPM)
```

In[164]:

```
for i in range(len(Y_pred_LPM)):
    if Y_pred_LPM.iloc[i] >= 0.5:
        Y_pred_LPM.iloc[i] = 1
    else:
        Y_pred_LPM.iloc[i] = 0


# In[165]:


conf_mat_LPM = confusion_matrix(Y_test, Y_pred_LPM)
conf_mat_LPM


# In[166]:


print('Test Sensitivity : ', conf_mat_LPM[1,1] / (conf_mat_LPM[1,1] + conf_mat_LPM[1,0]))
print('Test Specificity : ', conf_mat_LPM[0,0] / (conf_mat_LPM[0,0] + conf_mat_LPM[0,1]))
print('True Positive Rate : ', conf_mat_LPM[1,1] / (conf_mat_LPM[1,1] + conf_mat_LPM[1,0]))
print('False Positive Rate : ', 1 - (conf_mat_LPM[0,0] / (conf_mat_LPM[0,0] + conf_mat_LPM[0,1])))


# In[167]:


accuracy = (conf_mat_LPM[0, 0] + conf_mat_LPM[1, 1]) / np.sum(conf_mat_LPM)
accuracy


# In[168]:


print(classification_report(Y_test, Y_pred_LPM))


# # Probit Model

# ## Model Application

# In[169]:


X_Probit = X_train_scale.copy()


# In[170]:


Probit_1 = sm.Probit(Y_train, X_Probit)
result_Probit1 = Probit_1.fit()
result_Probit1.summary()


# ### Observation :: All varibles are significant
```

Variable Inflation Factor (VIF) calculation

In[171]:

```
calculate_vif(X_Probit)
```

Observation :: HumidityRatio is having high multicollinearity, so dropping it.

In[172]:

```
X_Probit = X_Probit.drop(['HumidityRatio'], axis = 1)
```

Building Another Model

In[173]:

```
Probit_2 = sm.Probit(Y_train, X_Probit)
result_Probit2 = Probit_2.fit()
result_Probit2.summary()
```

Observation : All variables are significant

Variable Inflation Factor (VIF) calculation

In[174]:

```
calculate_vif(X_Probit)
```

High multicollinearity between variables is absent

Final Probit Model

In[176]:

```
X_Probit_colnames = ['Temperature', 'Humidity', 'Light', 'CO2']
```

In[177]:

```
X_train_Probit = X_train.copy()
X_train_Probit = X_train_Probit[X_Probit_colnames]
X_train_Probit = sm.add_constant(X_train_Probit)
```

In[179]:

```
Probit_final = sm.Probit(Y_train, X_train_Probit)
result_Probit_final = Probit_final.fit()
result_Probit_final.summary()


# # Best fit Linear Probability model is,
# # Y = Φ(z)

# # z = 4.0622 - 0.4871 * Temperature + 0.0407 * Humidity + 0.0122 * Light + 0.0018 * CO2

# ## Evalution of Results

# ### 1. ROC Curve

# In[180]:


X_test_Probit = X_test.copy()
X_test_Probit = X_test_Probit[X_Probit_colnames]
X_test_Probit = sm.add_constant(X_test_Probit)


# In[181]:


Y_probit_probs = result_Probit_final.predict(X_test_Probit)
Y_probit_probs = Y_probit_probs.to_numpy()
Y_actual = Y_test.to_numpy()


# In[182]:


draw_roc_curve(Y_actual, Y_probit_probs)


# ### 2. Confusion Matrix

# In[183]:


Y_pred_Probit = result_Probit_final.predict(X_test_Probit)


# In[184]:


for i in range(len(Y_pred_Probit)):
    if Y_pred_Probit.iloc[i] >= 0.5:
        Y_pred_Probit.iloc[i] = 1
    else:
        Y_pred_Probit.iloc[i] = 0


# In[185]:
```

```
conf_mat_Probit = confusion_matrix(Y_test, Y_pred_Probit)
conf_mat_Probit


# In[186]:


print('Test Sensitivity : ', conf_mat_Probit[1,1] / (conf_mat_Probit[1,1] + conf_mat_Probit[1,0]))
print('Test Specificity : ', conf_mat_Probit[0,0] / (conf_mat_Probit[0,0] + conf_mat_Probit[0,1]))
print('True Positive Rate : ', conf_mat_Probit[1,1] / (conf_mat_Probit[1,1] + conf_mat_Probit[1,0]))
print('False Positive Rate : ', 1 - (conf_mat_Probit[0,0] / (conf_mat_Probit[0,0] + conf_mat_Probit[0,1])))


# In[187]:


accuracy = (conf_mat_Probit[0, 0] + conf_mat_Probit[1, 1]) / np.sum(conf_mat_Probit)
accuracy


# In[188]:


print(classification_report(Y_test, Y_pred_Probit))


# # Logit Model

# ## Model Application

# In[189]:


X_Logit = X_train_scale.copy()


# In[191]:


Logit_1 = sm.Logit(Y_train, X_Logit)
result_Logit1 = Logit_1.fit()
result_Logit1.summary()


# ### Observation :: All variables are significant

# ## Variable Inflation Factor (VIF) calculation

# In[192]:


calculate_vif(X_Logit)
```

Observation :: HumidityRatio is having high multicollinearity, so dropping it.

In[193]:

```
X_Logit = X_Logit.drop(['HumidityRatio'], axis = 1)
```

Building Another Model

In[194]:

```
Logit_2 = sm.Logit(Y_train, X_Logit)
result_Logit2 = Logit_2.fit()
result_Logit2.summary()
```

Observation : All variables are significant

Variable Inflation Factor (VIF) calculation

In[195]:

```
calculate_vif(X_Logit)
```

High multicollinearity between variables is absent

Final Logit Model

In[196]:

```
X_Logit_colnames = ['Temperature', 'Humidity', 'Light', 'CO2']
```

In[197]:

```
X_train_Logit = X_train.copy()
X_train_Logit = X_train_Logit[X_Logit_colnames]
X_train_Logit = sm.add_constant(X_train_Logit)
```

In[198]:

```
Logit_final = sm.Logit(Y_train, X_train_Logit)
result_Logit_final = Logit_final.fit()
result_Logit_final.summary()
```

Best fit Linear Probability model is,
Y = 1 / (1 + exp(-z))

z = 4.3288 - 0.7863 * Temperature + 0.0762 * Humidity + 0.0236 * Light + 0.0036 * CO2

Evalution of Results

1. ROC Curve

In[199]:

```
X_test_Logit = X_test.copy()
X_test_Logit = X_test_Logit[X_Logit_colnames]
X_test_Logit = sm.add_constant(X_test_Logit)
```

In[200]:

```
Y_logit_probs = result_Logit_final.predict(X_test_Logit)
Y_logit_probs = Y_logit_probs.to_numpy()
Y_actual = Y_test.to_numpy()
```

In[201]:

```
draw_roc_curve(Y_actual, Y_logit_probs)
```

2. Confusion Matrix

In[202]:

```
Y_pred_Logit = result_Logit_final.predict(X_test_Logit)
```

In[203]:

```
for i in range(len(Y_pred_Logit)):
    if Y_pred_Logit.iloc[i] >= 0.5:
        Y_pred_Logit.iloc[i] = 1
    else:
        Y_pred_Logit.iloc[i] = 0
```

In[204]:

```
conf_mat_Logit = confusion_matrix(Y_test, Y_pred_Logit)
conf_mat_Logit
```

In[205]:

```
print('Test Sensitivity : ', conf_mat_Logit[1,1] / (conf_mat_Logit[1,1] + conf_mat_Logit[1,0]))
print('Test Specificity : ', conf_mat_Logit[0,0] / (conf_mat_Logit[0,0] + conf_mat_Logit[0,1]))
print('True Positive Rate : ', conf_mat_Logit[1,1] / (conf_mat_Logit[1,1] + conf_mat_Logit[1,0]))
print('False Positive Rate : ', 1 - (conf_mat_Logit[0,0] / (conf_mat_Logit[0,0] + conf_mat_Logit[0,1])))


# In[206]:


accuracy = (conf_mat_Logit[0, 0] + conf_mat_Logit[1, 1]) / np.sum(conf_mat_Logit)
accuracy


# In[207]:


print(classification_report(Y_test, Y_pred_Logit))


# In[ ]:
```