Part 1

1)

	Negative	Positive
Negative	TN = 7	FP = 3
Positive	FN = 4	TP = 6

Precision :-

$$= TP / (TP + FP)$$
$$= 6 / (6 + 3)$$
$$= 6/9$$
$$= 0.67$$

Recall :-

$$TP/(TP+FN)$$

$$= 6/(6+4)$$

$$= 6/10$$

$$= 0.6$$

Accuracy :-

$$(TP+TN)/(TP+TN+FP+FN)$$

$$= (7+6)/(6+7+3+4)$$

$$= 13/20$$

$$= 0.65$$

F-measure :-

$$= \frac{2 \times ((\text{Precision} \times \text{recall})}{(\text{Precision} + \text{recall})}$$

$$= \frac{2 \times ((0.67 \times 0.6)}{(0.067 + 0.6)}$$

$$= 2 \times 0.3165$$

$$= \boxed{0.63}$$

2)

Regression Problem:

Initially, I uploaded the data into the Juypter notebook using the Pandas library. After that, I have done several pre-processing steps for the data, which includes univariate and bivariate analysis. Using the seaborne library, I created some scatter plots to check the data. Based on these plots, I concluded that houses with more convenience stores in the area and a low age have higher prices, whereas houses with a higher age have more MRT stations nearby and have lower prices. In the next step, I used boxplots to check if there were any outliers. To get a lower RMSE value, I applied the np.log () function for both train and test datasets. For the training and testing, I dropped the 'X1 transaction date' variable from both datasets as it is a dependent variable. I also dropped the 'No' variable as it was not relevant to the prediction. To train the model, I created two variables: **X2** (which contains the Train intendent variables) and **y2** (which contains the Train dependent variables). The shape of training datasets for models is now (301,5) & (301,). I have followed a similar method for the testing data for the model. The **X** variable contains Test independent variables, and the **y** variable contains Test dependent variables. The shape of test datasets for model is now (113,5) & (113,).

```
In [12]: X2 = train.loc[:,'X2 house age' : 'X6 longitude'] # Train Independent Variables
         y2 = train.loc[:,'Y house price of unit area'] #Train Dependent Variables

In [13]: X = test.loc[:,'X2 house age' : 'X6 longitude'] #Test Independent Variables
         y = test.loc[:,'Y house price of unit area'] #Test Dependent Variables

In [14]: print(X2.shape,X.shape)
         print(y2.shape,y.shape)

         (301, 5) (113, 5)
         (301,) (113,)
```

For the prediction, I chose **RandomForestRegressor**. For that I first imported **from sklearn.ensemble import RandomForestRegressor** library into the notebook and created model. Then I fitted the all the training independent and dependent data which I created earlier for the model. After that I predicted values for independent variable of Test dataset and saved those predicted values in y_predRF variable. **The RMSE (Root Mean Squared Error)** value of the model is approximately **0.214.**

```
from sklearn.metrics import mean_absolute_error, r2_score, mean_squared_error
from sklearn.ensemble import RandomForestRegressor

rfm = RandomForestRegressor()
rfm.fit(X2, y2)

y_predRF = rfm.predict(X)
r_squaredrfm = r2_score(y,y_predRF)*100


print('Algorithm : RandomForestRegressor')
print('MSE : ',mean_squared_error(y, y_predRF))
print('MAE :',mean_absolute_error(y, y_predRF))
print('RMSE :', np.sqrt(mean_squared_error(y, y_predRF)))
print('Accuracy :',rfm.score(X, y)*100)
print('Score :', r_squaredrfm)
```

```
Algorithm : RandomForestRegressor
MSE :  0.046094782575323186
MAE : 0.14336694847670806
RMSE : 0.21469695520738805
Accuracy : 66.42299609998756
Score : 66.42299609998756
```

I stored predicted values in _csv with the original values of test and compared both. I also used **pandas_profiling** library just for the summary of it. Here **0** is predicted values and **Y house price of unite area** is original values of test dataset.

First rows

	Y house price of unit area	0
0	3.148453	3.175262
1	3.983413	4.036240
2	3.772761	3.808968
3	3.475067	3.779419
4	3.765840	3.727861
5	3.758872	3.724100
6	3.246491	3.027333
7	3.768153	3.752772
8	3.139833	3.187275
9	3.538057	3.407163

Last rows

	Y house price of unit area	0
103	2.873565	2.864943
104	3.718438	3.673921
105	3.600048	3.695309
106	2.960105	2.985387
107	4.014580	3.871727
108	3.269569	3.388183
109	3.411148	3.645319
110	3.676301	3.780019
111	2.687847	2.742639
112	3.862833	3.695074

Classification Problem:

For the classification problem of the same dataset as asked in the problem set, two labelled has been created for the variable, **expensive and not-expensive** for train and test datasets.

```
train.loc[train['Y house price of unit area'] <= 30, 'Y house price of unit area'] = 'not-expensive'
```

```
train.loc[train['Y house price of unit area'] != 'not-expensive', 'Y house price of unit area'] = 'expensive'
```

```
test.loc[test['Y house price of unit area'] <= 30, 'Y house price of unit area'] = 'not-expensive'
```

```
test.loc[test['Y house price of unit area'] != 'not-expensive', 'Y house price of unit area'] = 'expensive'
```

```
train['Y house price of unit area'].value_counts()
```

```
expensive        207
not-expensive     94
Name: Y house price of unit area, dtype: int64
```

```
test['Y house price of unit area'].value_counts()
```

```
expensive        81
not-expensive    32
Name: Y house price of unit area, dtype: int64
```

After that I split the data for model and transformed the categorial values into numeric using **LabelEncoder.**

```
X2 = train.loc[:,'X2 house age' : 'X6 longitude'] # Train Independent Variables
y2 = train.loc[:,'Y house price of unit area'] #Train Dependent Variables
```

```
X = test.loc[:,'X2 house age' : 'X6 longitude'] #Test Independent Variables
y = test.loc[:,'Y house price of unit area'] #Test Dependent Variables
```

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
y2 = le.fit_transform(y2)
y = le.fit_transform(y)
```

Then using RandomForestClassifier library, I have created the random forest classification model to predict the value and fitted the relevant variables into it. After that, I predicted the values, and the **Accuracy** of the model is **88.495%** and **RMS**E values is **0.339.** I have also created confusion matrix and classification report for the model which you can see below.

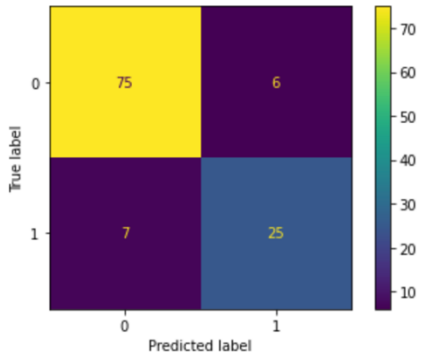
```
clf = RandomForestClassifier(n_estimators=500, max_depth=11)
clf.fit(X2, y2)
clf_prediction = clf.predict(X)
accuracy_rfs = accuracy_score(y, clf_prediction)
```

```
from sklearn.metrics import mean_absolute_error, r2_score, mean_squared_error
print('Accuracy :',accuracy_rfs*100)
print('RMSE :', np.sqrt(mean_squared_error(y,clf_prediction)) )
```

```
Accuracy : 88.49557522123894
RMSE : 0.3391817326856071
```

```
plot_confusion_matrix(clf, X, y)
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7fb5390c8eb0>
```




```
from sklearn.metrics import classification_report
print(classification_report(y,clf_prediction))
```

```
              precision    recall  f1-score   support

           0       0.91      0.93      0.92        81
           1       0.81      0.78      0.79        32

    accuracy                           0.88       113
   macro avg       0.86      0.85      0.86       113
weighted avg       0.88      0.88      0.88       113
```