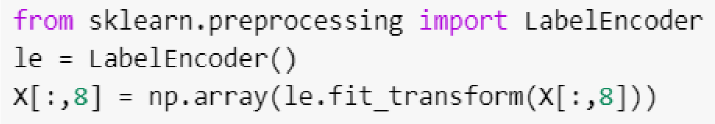
Assignment 3

Student ID: 21235117
Name: Diksha Srivastava
Course: Data Analytics (CSD1)

1. For the given task, I chose **Scikit-Learn** as:
   - Its documentation is easy to understand, and it is easy to implement as well especially for beginners.
   - It also provides API documentation; hence, it can be easily integrated with other platforms.
   - It provides us with most of the ML models like classification, regression, clustering, and many more.

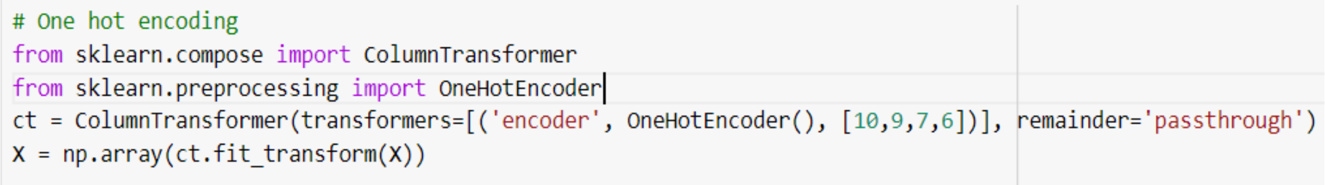
   Features of Scikit Learn:
   - It already comes with several built-in datasets like iris (for flowers), house prices, diabetes, and many more.
   - It already has packages for data pre-processing, dividing training and test set, feature scaling, Regression model, Decision Trees, KNN model, and many more.
   - It also provides functionality to visualize datasets, for example, doing cross-validation, creating learning curves etc.

2. Following pre-processing steps were performed before input to a model:
   - First, the txt file was converted to a CSV file by importing the txt in excel, delimited by space, and then save in CSV format. X, y is created such that X contained independent variables and y contained dependent variable.
   - Since the independent variable 'balcony' had categorical values yes/no, so, a Label Encoder was used to convert its values such that yes was converted to 1 and no was converted to 0.



```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
X[:,8] = np.array(le.fit_transform(X[:,8]))
```

   - Other categorical independent variables like type, ber, floor, and heating was converted into numerical values using one-hot encoding.



```
# One hot encoding
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(), [10,9,7,6])], remainder='passthrough')
X = np.array(ct.fit_transform(X))
```

3. **LINEAR REGRESSION**

   Linear regression is used to find linear relationship between two variables by fitting a linear equation to the given data. One is the explanatory variable, and the other is an independent variable.

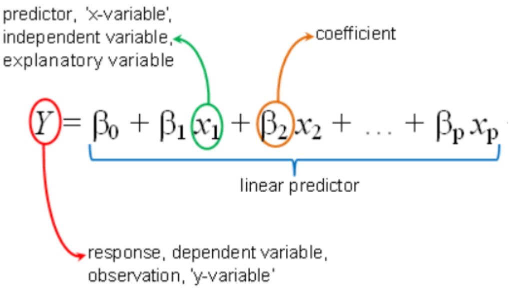
   It is a supervised machine learning model which finds the **best fit line/plane** to find the relationship between independent and dependent variable.

   TYPES OF LINEAR REGRESSION

   1. Simple: In simple linear regression, we try to find the relationship between one independent variable and one dependent variable. It can be represented by the following equation:

$$Y = \beta_0 + \beta_1 X$$

   2. Multiple: In multiple linear regression, we try to find the relationship between multiple independent variables and the corresponding dependent variable. It can be represented by the following equation:


predictor, 'x-variable', independent variable, explanatory variable
coefficient
$Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_p x_p$
linear predictor
response, dependent variable, observation, 'y-variable'

(Types of Linear Regression)

Which type we are using for our purpose?

We have to use multiple linear regression as we have 11 independent variables and 1 dependent variable.

Core of Linear Regression

- **Hypothesis:** A hypothesis shows the line/plane that represents the given dataset. It can be represented with the help of regression coefficients. A regression coefficient is associated with a feature which tells how important a feature is. The hypothesis for linear regression is of the following form:

  $$h_\theta(x) = \theta^T X$$

  where θ is the vector of regression coefficients and X is the vector of features.
- **Cost Function:** The objective of the cost function is to evaluate the errors the linear model is going to make so that we can develop a model with minimum error. It is represented as:

  $$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x_i) - y_i)^2$$

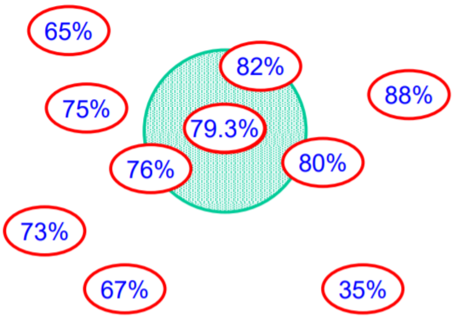
  Objective of solving would be to identify the values for parameters i.e. θ that will minimize the value of J(θ).

  (Cost Function)
- **Gradient descent:** It is an optimization technique that finds the optimal regression coefficients and minimizes the cost function. We iteratively find the gradient of the cost function at the current point and move in the opposite direction till a minimum cost error is achieved. It is represented as:

```
Repeat until convergence {
    $\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$
    (for j=0 to j=n) → (Update all $\theta_j$ s simultaneously before moving to next iteration. )
}
```

KNN FOR REGRESSION

K-nearest neighbour is a supervised machine learning approach in which the association between independent variables and the continuous outcome is found by calculating the average of observations in the same neighbourhood. The size of K should be taken in such a way that it minimizes the root mean-squared error.



Estimate: 79.3%

The k-nearest neighbour can be found by distance metrics such as Euclidean, Manhattan, or Minkowski.

(Distance Function)

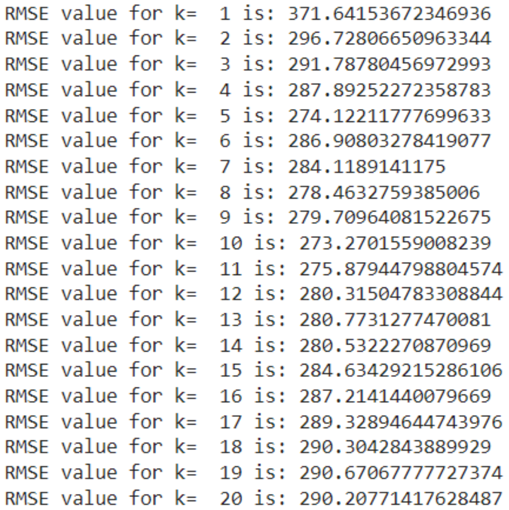
Euclidean	$\sqrt{\sum_{i=1}^{k} (x_i - y_i)^2}$
Manhattan	$\sum_{i=1}^{k} \lvert x_i - y_i \rvert$
Minkowski	$\left(\sum_{i=1}^{k} (\lvert x_i - y_i \rvert)^q \right)^{1/q}$

Here, k is the number of features/attributes to be used to calculate the distance, x and y are vectors containing the values of the attributes we are interested in.

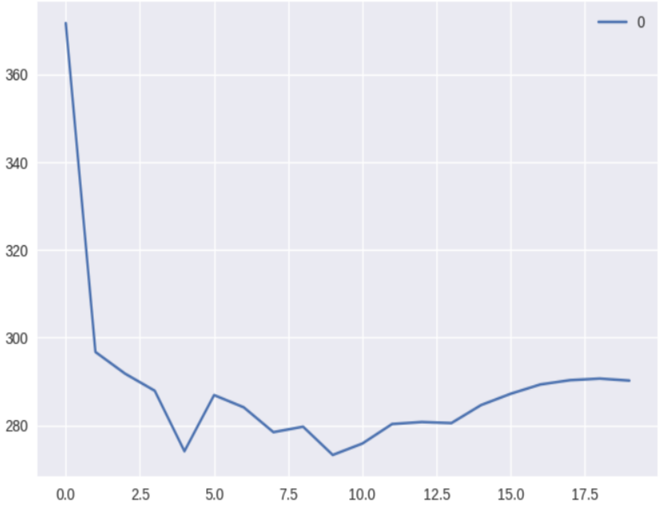
TYPES OF KNN REGRESSION

1. Uniform weighting: It assumes that each neighbour is given equal weighting, Thus, the predicted value is simply the average of k-nearest neighbours.
2. Distance weighting: It assumes that each neighbour is weighted based on the distance from the query. It is the inverse square of distance from the query.

4. **KNN Parameter Settings:** While developing the KNN model, it was important to choose the appropriate k value in order to train the model.



```
RMSE value for k=  1 is: 371.64153672346936
RMSE value for k=  2 is: 296.72806650963344
RMSE value for k=  3 is: 291.78780456972993
RMSE value for k=  4 is: 287.89252272358783
RMSE value for k=  5 is: 274.12211777699633
RMSE value for k=  6 is: 286.90803278419077
RMSE value for k=  7 is: 284.1189141175
RMSE value for k=  8 is: 278.4632759385006
RMSE value for k=  9 is: 279.70964081522675
RMSE value for k=  10 is: 273.2701559008239
RMSE value for k=  11 is: 275.87944798804574
RMSE value for k=  12 is: 280.31504783308844
RMSE value for k=  13 is: 280.7731277470081
RMSE value for k=  14 is: 280.5322270870969
RMSE value for k=  15 is: 284.63429215286106
RMSE value for k=  16 is: 287.2141440079669
RMSE value for k=  17 is: 289.32894644743976
RMSE value for k=  18 is: 290.3042843889929
RMSE value for k=  19 is: 290.67067777727374
RMSE value for k=  20 is: 290.20771417628487
```



After training the model for k value 1 to 20, it was found that the RMSE for k = 10 was minimum i.e., 273.27 and thus, k = 10 was chosen to train the model and make predictions. The corresponding graph was also plotted between RMSE and different k values.

LINEAR REGRESSION Parameter Settings: In order to train and fit the data to the model, some parameters play an important role in order to achieve convergence in Linear regression. The parameters are as follows:

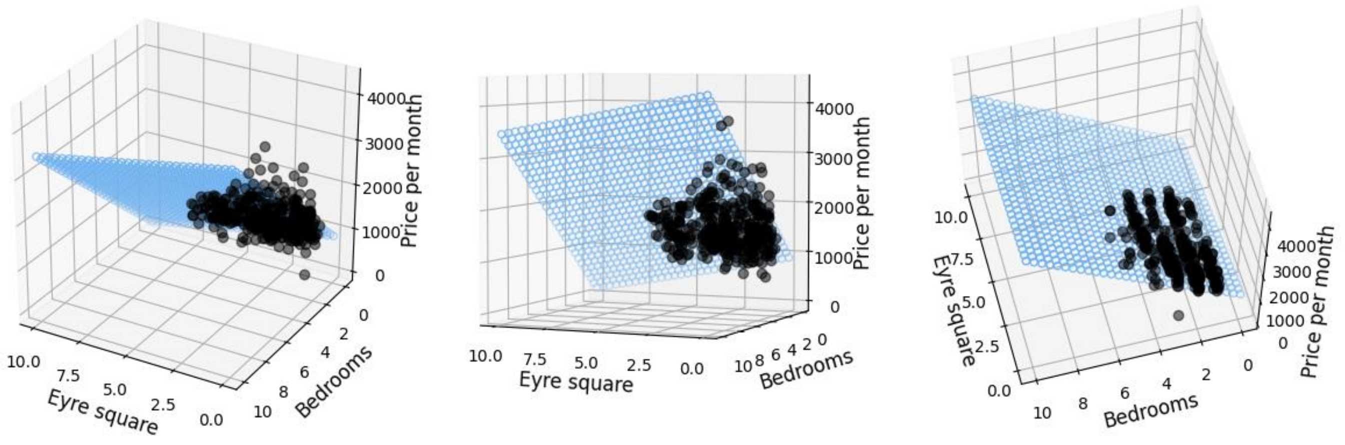
Learning Rate: It determines the magnitude of amount to move in gradient descent. The change in coefficient is learning rate times the gradient. If it is too small, the algorithm will converge slowly, but if it is too large, it can lead to non-convergence i.e., bouncing back and forth between the convex function and never reaching the local minimum. For training the linear regressor model, a learning rate of **0.01** was chosen, it was done so because it is neither too high nor too low, hence, reaching a convergence would be easy.

Tolerance Factor: It is the stopping criterion. The value used is **0.001**. Having a tolerance factor ensures that the minimum cost is found even if the convergence is not reached. The training will stop if the difference in cost is less than the tolerance level. A small value is chosen for this so that if convergence is achieved before this then that coefficient is considered.

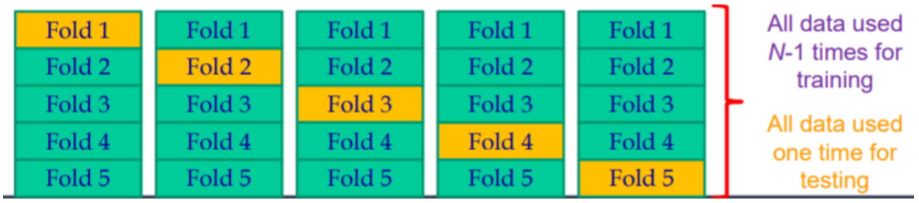
Regularization Factor: Regularization is encouraging zero/low values for regression coefficients which are not important. It can be helpful when we have a lot of features. We add a term to cost function to reflect some measure of complexity of the hypothesis. Its value is set to **0.0001.** The higher its value, the higher the regularization factor. Thus, for our case, it is set to a smaller value.

Fit intercept: It is set to **True**. It tells whether the intercept should be estimated or not. If this value is set to false, then the intercept is not estimated, and the data is assumed to be already centred.

After training the linear regression model, the following best fit plane was created. The plane was created between price per month ~ (distance from Eyre square, number of bedrooms).



5. The dataset was divided into training set, validation set, and test set.
    - The training and test set was divided randomly in such a way so that 1/3rd of the data was the test set, and the rest was training set.
    - From the training set, a validation set was created using **Cross-Validation** technique. A **5-fold cross validation** was used in which the model was trained 5 times. Each time the dataset was divided into 5 folds such that 1-fold was used for validation set and the other 4 folds was used for the training set.
    - The model was trained on the 4 folds training set and then tested on the validation set such that all data is used for testing once. Further, the training and validation set is combined and tested on the held-out test set.



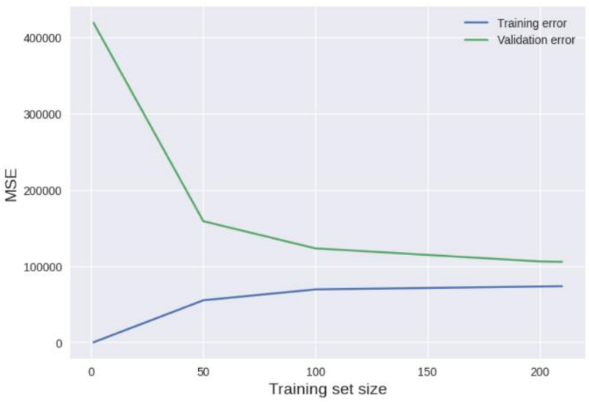
Fold 1
Fold 2
Fold 3
Fold 4
Fold 5
All data used *N*-1 times for training
All data used one time for testing

To check overfitting and underfitting, the RMSE value was observed 5 times for the training set and the test set when the model was trained.

- If the RMSE is low for training set and high for test set, then that means model is overfit.
- If the RMSE is high for training set, then that means model is underfit.

Over-fitting and under-fitting for Linear regression and KNN

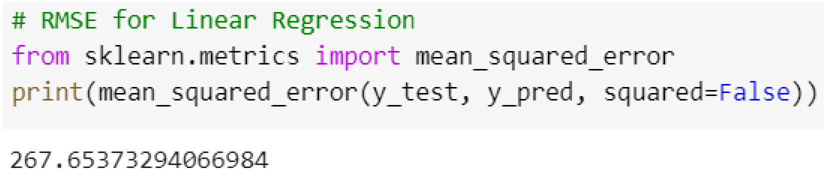
- Linear Regression generally have high bias and low variance as it makes linearity assumption and thus a risk of underfitting is there. A learning curve showing the error rate on training and test set is also shown. _(Learning Curve)
- For KNN, if k is too low, then the model tends to overfit data if the data is noisy. However, if k is too high, then the model tends to underfit.



6. **EVALUATION**

RMSE is one of the evaluation metrics which can be used for regression task. It measures how concentrated the data is around the line of best fit. The less the RMSE the better the model is.

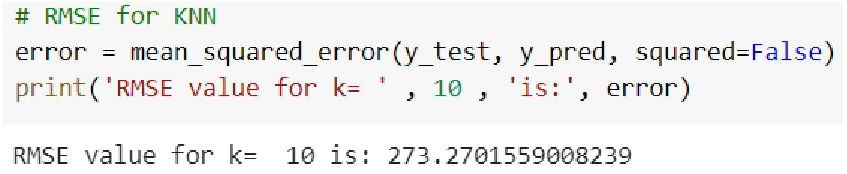
RMSE for multiple linear regression is:



```
# RMSE for Linear Regression
from sklearn.metrics import mean_squared_error
print(mean_squared_error(y_test, y_pred, squared=False))

267.65373294066984
```

RMSE for KNN regressor is:



```
# RMSE for KNN
error = mean_squared_error(y_test, y_pred, squared=False)
print('RMSE value for k= ' , 10 , 'is:', error)

RMSE value for k=  10 is: 273.2701559008239
```

The two regression models showed almost similar result. However, Multiple Linear regressor performed slightly better than KNN as its RMSE value is less than the value for KNN. This is so because, Linear Regression made use of all the training data to create a best-fit plane that represents the given dataset. The best fit plane was found by finding the optimal values of regression coefficients such that cost is minimum. The predictions are then made by finding the point on the best-fit plane. However, in case of KNN, a k value is selected and then the average of those k nearest neighbours is taken. All the training set is not considered but only the k nearest neighbours is selected and the predictions are done. Hence, the result is different in both models.

References

Cost Function. n.d. https://www.crayondata.com/machine-learning-linear-regression-gradient-descent-part-1/

Distance Function. n.d. https://www.saedsayad.com/k_nearest_neighbors_reg.htm

Learning Curve. n.d. https://www.dataquest.io/blog/learning-curves-machine-learning/

Types of Linear Regression. n.d. https://medium.datadriveninvestor.com/types-of-linear-regression-89f3bef3a0c7