Capstone Project

Big Data Analysis: Machine Learning Model for Predicting House Price and Type in Melbourne through previous data.

**Submitted by:** 1. NAME: **Vignya Durvasula**

ROLL NO.: **20P61A6615**

2. NAME: **Shreejit**

ROLL NO.: **20P61A6647**

Bachelors of Technology in Artificial Intelligence and Machine Learning

College name: Vignana Bharathi Institute of Technology

College Advisor: Prof. K Shirisha

**Submitted to**: YBI foundation, Delhi.

April 2022

**ABSTRACT**

The field of Machine Learning is the latest art of communication between Human Intelligence and Artificial Intelligence in heuristic ways. In simpler words, enabling computers make successful predictions using past experiences and has exhibited an impressive development recently with the help of the rapid increase in the storage capacity and operating speeds of computers is called Machine Learning. Machine Learning methods have been widely employed in every single domain of the world. The ease and cost of analyses have led to the development of sophisticated machine learning approaches in many sectors. In this project, we create a perfect working model of House Price Prediction using the concepts of supervised learning, a branch of Machine Learning. We start with looking at the Challenges and Difficulties faced, understand the complete code and finally, we also test our model and evaluate the performance by taking a sample prediction.

**List of Abbreviations, Figures and Important terms**

**Figures:**

Figure 1: AI vs ML vs DL

Figure 2: Types of Machine Learning

Figure 3: Types of Encoding Technique.

**Abbreviations:**

**AI:** Artificial Intelligence

**ML:** Machine Learning

**DL:** Deep Learning

**KNN:** K-Nearest Neighbour

**SVM:** Support Vector Machine

**Important Terms:**

**1. Attribute:** Attributes are the items of data that are used in machine learning. Attributes are also referred as variables, fields, features or predictors.

**2. Target:** The target variable of a dataset is the feature or attribute of a dataset about which you want to gain a deeper understanding or whose value one wants to predict

**3. Regression Analysis:** Regression is a supervised ML technique for investigating the linear relationship between independent variables or features and a continuous dependent variable or outcome.

**4. Classification Analysis:** Classification is a supervised ML technique that is used to categorize the categorical labelled data into classes or groups and predict a categorical result for input data.

**NOTE: Important terms elsewhere are bolded for better understanding.**

**Table of Contents**

1. Abstract 2

2. List of abbreviations, figures and Important terms 3

3. Chapter 1: Introduction 5

4. Chapter 2: Data Source, Properties, Challenges and Libraries 10

5. Chapter 3: Modelling Approach 17

6. Chapter 4: Algorithm 23

7. Chapter 5: Results and Conclusion 33

8. References 35

**CHAPTER – 1**

**INTRODUCTION**

There is this famous saying, “Home is where the Heart is”. If houses are made with brick and mortar, then a Home is made with expectations and desires. These expectations or desires when put on a paper come out to be the adjectives that essentially describe a house. We being under the Machine Learning domain, refer to these adjectives as **attributes** or **features.** Every house is hence obliged to have all these attributes as its characteristics as an individual wishes it to be for making it a happy place to live.

The **values** for these attributes may vary from record to record (or tuple to tuple) because the odds of having same tastes and likes is quite high. However, these values are bound to be in a **particular range of numbers or finite number of categories** with some exceptions. These values may be numbers, characters, strings, date or of **any** other **datatype** according to the attribute we are talking about. For an instance, Address attribute of a house holds strings(‘object’) whereas, on the other hand, Price holds ‘float32’ or ‘double64’ value in it. Although, all the **values under** **one attribute** should be of the **same type.**

Its rather obvious that the house ‘Price’ is **dependent** on the values of other attributes. For example, A 3BHK is often costlier than a 2BHK also, A 3BHK in a developing area is sometimes more reasonable compared to a 2BHK in a developed or an urban area. In contrast to, if we talk about the ‘Address’ or ‘S no.’ attributes, one cannot say its dependent on any other attribute. Therefore, it’s fairly evident that our data contains some **dependent attributes** and some **independent attributes**.

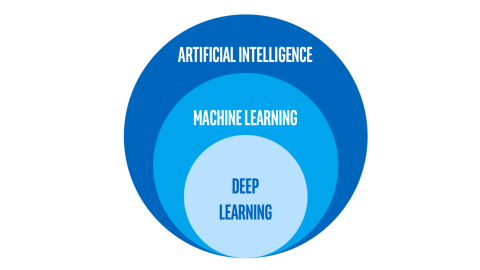


Figure 1: AI vs ML vs DL

Getting into the technical aspects, **Artificial intelligence [A.I.]** is a technology which enables a machine to **simulate and mimic human behaviour** whereas, **Machine Learning [M.L.]** is a subfield of AI which allows a machine to **automatically learn** from past data and allow software applications to become more **accurate at predicting outcomes without being explicitly programmed** to do so. The goal of AI is to make a smart computer system that think and work like humans to solve complex problems. A possible way to achieve this goal is from the ML and sub branches of ML. It is basically the study of computer algorithms that can directly or indirectly accelerate automatically through experience and training.

Machine Learning can widely be classified into **three main types** that deal with different and unique set of paradigms.

Three types of ML are:

* Supervised Learning
  + Regression
  + Classification
* Unsupervised Learning
  + Clustering
  + Association
* Reinforcement Learning
  + Decision Making

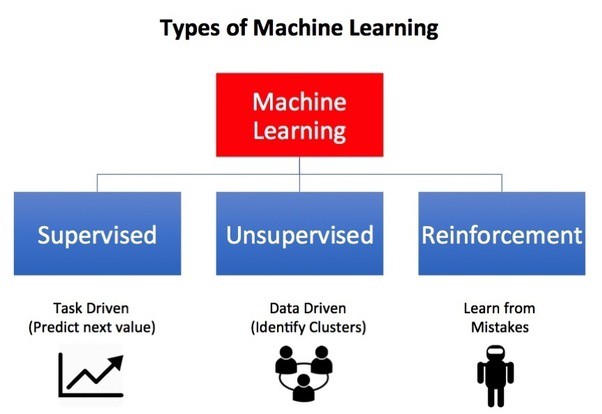


Figure 2: Types of Machine Learning

**I. Supervised Learning**:

Supervised learning is the most popular paradigm for machine learning. It is the easiest to understand and the simplest to implement. In this type of Learning, the data is in the form of Labels hence it is called as Labelled or Structured Data. This Labelled Data is now fed to the algorithm in which the model is prepared. Overtime the algorithm will learn to approximate the actual nature of relationship between the available labels. A fully trained Supervised Learning algorithm is capable of predicting a target value when the attribute values are provided. It is very similar to teaching a child with the use of flash cards. It is a **task-oriented algorithm**. Accuracy of prediction depends upon the number of tuples or records the algorithm is trained on or fed with. It mainly deals with the Regression and Classification Problems.

**Examples**: Spam Classifiers and Face Recognition systems.

**II. Unsupervised Learning:**

**Unsupervised learning** is pretty much the opposite of Supervised Learning. In this type of Learning, the data is not in the form of Labels hence the data is called as Unlabelled or Unstructured Data. Unsupervised Learning is a ML technique that uses AI algorithms to train and finally organise unstructured data into clusters by identifying the patterns in the dataset with the help of the data points. It is similar to asking a kid to separate apples and mangoes from a basket of them. It is a **property - oriented technique**. An overwhelming majority of data in the world is unlabelled which clearly says that unsupervised learning technique is one of the most conventionally used ML technique to deal with namely Clustering and Association Problems.

**Examples**: Recommendation Systems.

**III. Reinforcement Learning:**

**Reinforcement learning** is relatively different when compared to supervised and unsupervised learning. Where we can easily see the relationship between supervised and unsupervised (the presence or absence of labels), the relationship to reinforcement learning is a bit murkier. Trying to tie reinforcement learning closer to the other two by describing it as a type of learning that relies on a time-dependent sequence of labels is a common practice, nevertheless, my opinion is that it simply makes things more confusing. Preferring to look at reinforcement learning as a model learning from its mistakes is comparatively better for understanding. Place a reinforcement learning algorithm into any environment and it will make a lot of mistakes in the beginning. So long as we provide some sort of signal to the algorithm that associates good behaviours with a positive signal and bad behaviours with a negative one, we can reinforce our algorithm to prefer good behaviours over bad ones. Over time, our learning algorithm learns to make less mistakes than it used to. It is basically a **behaviour driven model** that is largely used to deal with Decision-Making Problems.

**Examples:** Video games, Warcraft, Resource Management.

In this Project, we are going to look at a Big Data Analysis and a model predicting the ‘Price’ (**Regression**) and ‘Type’ (**Classification**) of a house upon the **Melbourne Housing Market** dataset.

**CHAPTER – 2**

**2.1: DATA SOURCE**

**Dataset name**: Melbourne Housing Market

**Source**: One of the many datasets in GitHub profile of YBI Foundation.

**2.2: PROPERTIES**

**Storage occupied**: 2.1MB

**Type of file**: Comma Separated values (.csv)

**No. of Attributes**: 21

**No. of Records**: 13580

**Unique no. values under each Attribute**:

* Suburb 314
* Address 13378
* Rooms 9
* Type 3
* Price 2204
* Method 5
* SellerG 268
* Date 58
* Distance 202
* Postcode 198
* Bedroom2 12
* Bathroom 9
* Car 11
* Landsize 1448
* BuildingArea 602
* YearBuilt 144
* CouncilArea 33
* Lattitude 6503
* Longtitude 7063
* Regionname 8
* Propertycount 311

**Different Data types**: datetime64[ns](1), float64(12), int64(1), object(6)

**2.3: CHALLENGES**

This particular data set being a Big Data we get to see a plethora of challenges:

**I. Missing Data:**

**Missing data or Missing values** is one of first problems that while pre-processing is to be handled with utmost care before working on a dataset. Missing data in a Dataset is generally represented as “NaN” which is nothing but Not a Number or “inf” which means infinite. These two values (Nan and inf) along with blank spaces comprise to form missing data in a dataset. There two known ways to handle missing value:

1. Firstly, the manual way, we can handle missing data using the mean of any other column or attribute:

Syntax:

df[‘Missingdatacolumnname’].fillna(df.groupby([‘HelpingAttribute’])[‘Missingdatacolumnname’].transform(‘mean’), inplace=True)

where, df means dataframe

fillna: built in python function used to fill data

inplace=True is used to change the default behaviour

Note: in place of “mean”, “median” can also be used if the user wants to fill the values by frequency than average.

2. Second method, the automatic method is by using the KNN imputer. The k-Nearest-Neighbour imputer is used to impute or predict each sample’s missing values. This method also works well for solving our problem.

Syntax:

from sklearn.impute import KNNImputer

imputer = KNNImputer(n\_neighbors=2)

**In this project the first or the manual method is being used.**

**II. Encoding:**

Encoding is nothing but the process of converting a user readable language to machine readable form. Most of the values in the data set are in the machine-readable form i.e., int64, float64 but to operate on the dataset all of the attributes must be read and understood by the machine. The main problem comes with the object type values which are nothing but strings. Strings are stored as Unicode in python which means each character in the string is represented with a code point.  So, each string is just a sequence of Unicode code points. For efficient storage of these strings, the sequence of code points is converted into a set of bytes. The process is conventionally referred to as Encoding.

Types of Encoding techniques:

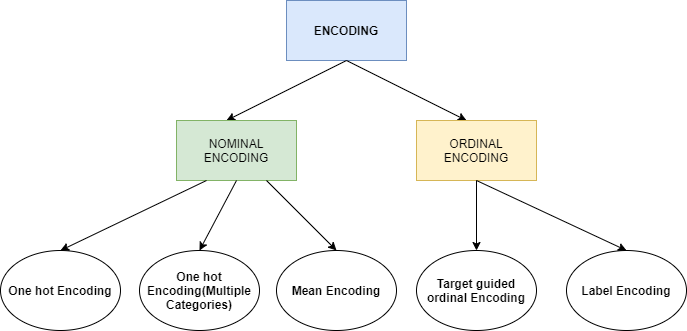


Figure 3: Types of Encoding Techniques

**In this project, Label Encoding technique has been used to encode attributes namely ‘Method’, ‘Type’ and ‘Regionname’ as the data to be encoded is fairly Ordinal.**

**III. Scaling:**

It’s kind of a manifest that any large dataset has values of different scales or units. For an instance ‘Landsize’ attribute has its values in hundreds or thousands where as ‘Price’ attribute has its values in millions. Moreover, the value ‘300cm’ can also be represented as ‘3m’. None of the above representations are wrong or flawed, both are right. So how are we supposed to represent? This might seem very natural or normal to us humans but when a machine is dealing with the dataset on the whole it is a requisite to deal with this problem and scale all values to be having same units before the dataset is operated on. A ML model is all about prediction and accuracy, even without scaling the data the model will predict a value but the accuracy of the value is affected. Furthermore, a model with high accuracy may also change its behaviour after deployment as and when the user data is being increased by time. These affects maybe positive or negative.

**In this project Standard Scaling has been put into use.**

Syntax:

from sklearn.preprocessing import StandardScaler

s=StandardScaler()

X\_train\_s=s.fit\_transform(X\_train)

X\_test\_s=s.fit\_transform(X\_test)

This scales the independent data in the dataset.

**IV. Oversampling and Under-sampling of data:**

**Oversampling** refers to randomly selecting examples from the minority class, with replacement, and adding them to the training dataset whereas on the other hand, **Under-sampling** is the process of randomly selecting examples from the majority class and deleting them from the training dataset. The concept of Oversampling and Under-sampling is largely used to compensate for an imbalance that is either already present in the data, or likely to develop if a purely random sample were taken.

Syntax:

For oversampling:

from imblearn.over\_sampling import RandomOverSampler

r=RandomOverSampler(random\_state=2408)

For under-sampling:

from imblearn.under\_sampling import RandomUnderSampler

r=RandomUnderSampler(random\_state=2408)

**2.4: LIBRARIES**

1. **Pandas**: **Python Data Analysis** Libraryused for data manipulation and data analysis.

2. **Numpy**: **Numerical Python** library used to work with arrays, linear algebra, matrices etc.

3. **Seaborn**: used for Statistical Data Visualisation

4. **Matplotlib.pyplot**: Numerical Data Visualisation.

5. from **imblearn.over\_sampling** import **RandomOverSampler**: for Oversampling

6. from **imblearn.under\_sampling** import **RandomUnderSampler**: for Under-Sampling

7. from **sklearn.model\_selection** import **train\_test\_split**: for splitting data into train and test data.

8. from **sklearn.preprocessing** import **StandardScaler**: for scaling the data

9. Libraries for creating models:

For Regression:

* from **sklearn.linear\_model** import **LinearRegression**
* from **sklearn.neighbors** import **KNeighborsRegressor**
* from **sklearn.tree** import **DecisionTreeRegresso**
* from **sklearn.ensemble** import **RandomForestRegressor**

For Classification:

* from **sklearn.linear\_model** import **LogisticRegression**
* from **sklearn.neighbors** import **KNeighborsClassifier**
* from **sklearn.tree** import **DecisionTreeClassifier**
* from **sklearn.svm** import **SVC** #Support Vector Machine

10. For Accuracy:

For Regression: from **sklearn.metrics** import **mean\_absolute\_percentage\_error,mean\_absolute\_error,r2\_score**

For Classification:

from **sklearn.metrics** import **confusion\_matrix, classification\_report**

**CHAPTER 3**

**MODELLING APPROACH**

Since we are dealing with both Regression analysis and Classification analysis there are 8 types of models where each type has 4 models in them. There are various other models but we are only going to look at these types as they are not only easy to understand but also efficient. The list of all the models we are going to look at are as follows:

**Regression Analysis:**

* + - * + Linear Regression
        + KNN
        + Decision Tree
        + Random Forest

**Classification Analysis**:

* + - * + Logistic Regression
        + KNN
        + Decision Tree
        + Support Vector Machine

Let’s look at each one of them in detail:

**I. Regression Analysis**

**I.1. Linear Regression:**

**Linear Regression** is one of the easiest and most widely used Regression model of Supervised Machine Learning. It basically uses Statistical Methods to perform the Predictive Analysis. It works by showing the linear relationship between the y-dependent variable and X-independent variables. Linear relationship is nothing but how the value of dependent variable changes with the change in the value of independent variable. It provides a sloped straight line representing he relationship between the variables.

Mathematically,

y = a + bX + ε

where, y: dependent variable or target variable

X: independent variable or features

a: intercept of the line

b: Linear regression coefficient

ε: Random error

There are two main types of Linear regression

* **Simple Linear Regression**: If a single independent variable is used
* **Multiple Linear Regression**: if more than one independent variables are used

**I.2. & II.2. KNN Regressor/Classifier:**

KNN stands for **K-Nearest Neighbour**. KNN regression is a non-parametric method of Supervised Machine Learning, which in an intuitive manner, approximates the association between independent variables and the continuous outcome by averaging the observations in the same neighbourhood. The size of the neighbourhood needs to be set by the analyst or can be chosen using cross-validation to select the size that minimises the mean-squared error or the accuracy. While the method is quite appealing, it quickly becomes impractical when the dimension increases, viz., when there are many independent variables. The number of nearest neighbours to a new unknown variable that has to be predicted or classified is denoted by the symbol 'K'. It is a Distance-based ML algorithm.

**I.3. & II.3. Decision Tree Regressor/Classifier:**

**Decision trees** use multiple algorithms to decide to split a node into two or more sub-nodes. The creation of sub-nodes increases the homogeneity of resultant sub-nodes. In other words, we can say that the purity of the node increases with respect to the target variable. The decision tree splits the nodes on all available variables and then selects the split which results in most homogeneous sub-nodes. The algorithm selection is also based on the type of target variables. The decision of making strategic splits heavily affects a tree’s accuracy. **A decision tree simply asks a question and based on the answer (Yes/No), it further split the tree into subtrees.** The decision criteria are different for classification and regression trees. The most commonly used decision tree algorithms are:

* **ID3** → extension of D3
* **C4.5** → successor of ID3
* **CART** → Classification and Regression Tree
* **CHAID** → Chi-square automatic interaction detection Performs multi-level splits when computing classification trees
* **MARS** → multivariate adaptive regression splines

**I.4. Random Forest Regressor/Classifier:**

**Random forest** is a Supervised Machine Learning Algorithm that is used widely in Classification and Regression problems. It builds decision trees on different samples and takes their majority vote for classification and average in case of regression.

One of the most important features of the Random Forest Algorithm is that it can handle the data set containing continuous variables as in the case of regression and categorical variables as in the case of classification. It performs better results for classification problems. There are two main types of Random Forest methods that put into practice

1. **Bagging**– It creates a different training subset from sample training data with replacement & the final output is based on majority voting. For example, Random Forest.

2. **Boosting**– It combines weak learners into strong learners by creating sequential models such that the final model has the highest accuracy. For example, ADA BOOST, XG BOOST.

In this project Random Forest is only used for Regression Problem.

**Difference between Decision tree and Random Forest:**

|  |  |  |
| --- | --- | --- |
| Sno. | Decision Tree | Random Forest |
| 1. | Decision trees normally suffer from the problem of overfitting if it’s allowed to grow without any control. | Random forests are created from subsets of data and the final output is based on average or majority ranking and hence the problem of overfitting is taken care of. |
| 2. | A single decision tree is faster in computation. | It is comparatively slower. |
| 3. | When a data set with features is taken as input by a decision tree it will formulate some set of rules to do prediction. | Random forest randomly selects observations, builds a decision tree and the average result is taken. It doesn’t use any set of formulas. |

**II. Classification Analysis:**

**II.1. Logistic Regression:**

**Logistic regression** is a statistical analysis method to predict a binary outcome, such as yes or no, based on prior observations of a data set. It predicts a dependent data variable by analysing the relationship between one or more existing independent variables. It can take into consideration multiple input criteria. Logistic regression has become an important tool in the discipline of machine learning. It allows algorithms used in machine learning applications to classify incoming data based on historical data. As additional relevant data comes in, the algorithms get better at predicting classifications within data sets. Logistic regression can also play a role in data preparation activities by allowing data sets to be put into specifically predefined buckets during the extract, transform, load process in order to stage the information for analysis. It mainly streamlines the mathematics for measuring the impact of multiple variables with a given outcome. The resulting models can help tease apart the relative effectiveness of various interventions for different categories of people, such as “young or old” or “male or female”. They can also transform raw data streams to create features for other types of AI and ML techniques. In fact, logistic regression is one of the commonly used algorithms in machine learning for binary classification problems, which are problems with two class values, including predictions such as **"this or that," "yes or no," and "A or B."** It can also estimate the probabilities of events, including determining a relationship between features and the probabilities of outcomes.

**II.4. Support Vector Machine:**

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning. The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane. It chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as **Support Vector Machine**. In this Project SVM is only used for Classification Problem.

**Types of SVM:**

SVM can be of two types:

* **Linear SVM**: Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier.
* **Non-linear SVM**: Non-Linear SVM is used for non-linearly separated data, which means if a dataset cannot be classified by using a straight line, then such data is termed as non-linear data and classifier used is called as Non-linear SVM classifier.

**CHAPTER 4**

**ALGORITHM**

**The code is written using the basic format and is being explained in the form of comments as and where needed.**

**Code:**

**Data ingestion and preparation:**

**Step 1: Import Libraries**

import pandas as pd#for data manipulation

import numpy as np#for working with arrays in the dataset

import seaborn as sns#for statistical data visualisation

import matplotlib.pyplot as plt#for dynamic data visualisation

**Step 2: Import Dataset**

df=pd.read\_csv('https://github.com/ybifoundation/Dataset/blob/main/Melbourne%20Housing%20Market.csv?raw=true')#importing the dataset as a .csv file

df#displaying the dataframe

**Step 3: Data Pre-processing**

df.shape#shape builtin function is used to know the total number of rows and columns in the dataset

df.nunique()#represents the number of unique values under each attribute

df.columns#displaying column names

df.info()#displays the basic information of the dataset

df['Date']=pd.to\_datetime(df['Date'])#converting the date attribute from object datatype to date datatype

**Step 4: Adjusting Missing values**

df.drop('CouncilArea',axis='columns', inplace=True)#Because its of object datatype moreover we already many other attributes that refer to the address and hence can be considered as subsidiary

df['BuildingArea'].fillna(df.groupby(['Type'])['BuildingArea'].transform('mean'), inplace=True)#filling the missing values in a particular attribute using the mean of other attributes

df['YearBuilt'].fillna(df.groupby(['Type'])['YearBuilt'].transform('mean'), inplace=True)

df['Car'].fillna(df.groupby(['Type'])['Car'].transform('mean'), inplace=True)

df.info()#One can observe that the missing values are all now filled up

df.describe()#displaying description of the data in the DataFrame

df.nunique()

**Step 5: Encoding:**

df['Method'].value\_counts()#displaying the number of unique values under each label of the attribute

df.replace({'Method':{'S':1,'SP':2,'PI':2,'VB':2,'SA':2}},inplace=True)#grouping

df['Method'].value\_counts()#one can observe the change in label and value count

df['Type'].value\_counts()

df.replace({'Type':{'h':1,'u':2,'t':2}},inplace=True)

df['Type'].value\_counts()

df['Regionname'].value\_counts()

df.replace({'Regionname':{'Southern Metropolitan':1,'Northern Metropolitan':2,'Western Metropolitan':3,'Eastern Metropolitan':3,'South-Eastern Metropolitan':3,'Eastern Victoria':3,'Northern Victoria':3,'Western Victoria':3}},inplace=True)

df['Regionname'].value\_counts()

**Representation of Encoded dataset and information**

df

df.info()

df.describe()

df.nunique()

**Data Visualisation**

sns.pairplot(df)#visualisation study

**Regression Analysis:**

**Step 1: Define X and y**

y=df['Price']#'Price' is our Regression target

np.log(y)#for increasing efficiency

X=df.drop(['Suburb','Address','SellerG','Date', ‘Price’],axis=1)

**Dealing with Oversampling of Data**

**‘Price’**

from imblearn.over\_sampling import RandomOverSampler

r=RandomOverSampler(random\_state=2408)

**Before Oversampling**

X.shape, y.shape

y.value\_counts()

X.value\_counts()

**After Oversampling**

X, y = r.fit\_resample(X,y)

X.shape, y.shape

y.value\_counts()

X.value\_counts()

**Step 2: Splitting Data**

from sklearn.model\_selection import train\_test\_split

X\_train,X\_test,y\_train,y\_test=train\_test\_split(X,y,test\_size=0.3,random\_state=2529)#since test size is given as 30%, train size is 70%

X\_train.shape,X\_test.shape,y\_train.shape,y\_test.shape

**Scaling the dataset**

from sklearn.preprocessing import StandardScaler

s=StandardScaler()

X\_train\_s=s.fit\_transform(X\_train)#Scaling train data

X\_test\_s=s.fit\_transform(X\_test)#Scaling test data

**Visualisation and Impact of Scaling**

* Reduced impact of outliers

X\_train\_s=pd.DataFrame(X\_train\_s,columns=X\_train.columns)

X\_test\_s=pd.DataFrame(X\_test\_s,columns=X\_test.columns)

fig, (ax1, ax2) = plt.subplots(ncols=2, figsize=(12, 5))

ax1.scatter(X\_train['YearBuilt'], X\_train['Landsize'],color='green')

ax1.set\_title("Before Scaling")

ax2.scatter (X\_train\_s['YearBuilt'], X\_train\_s['Landsize'],color="red")

ax2.set\_title("After Scaling")

plt.show()

**Note: Trying this code on different attributes will help you understand the impact of scaling**

**Step 3: Creating Model: 4 models of Regression**

**Model 1: Linear Regression Model**

from sklearn.linear\_model import LinearRegression

model=LinearRegression()#Linear Regression Model

**Model 2: KNN Model**

from sklearn.neighbors import KNeighborsRegressor

model=KNeighborsRegressor()#K-Nearest Neighbour Model

**Model 3: Decision Tree Model**

from sklearn.tree import DecisionTreeRegressor

model=DecisionTreeRegressor()#Decision tree Model

**Model 4: Random Forest Regressor**

from sklearn.ensemble import RandomForestRegressor

model=RandomForestRegressor()#Random Forest Model

**Note: Any one of the for models can be chosen. Accuracies vary according to the model that is chose.**

**Step 4: Training the Model**

model.fit(X\_train\_s,y\_train)#model that is used in the previous step is being trained in this step

**Step 5: Predicting Model**

y\_pred=model.predict(X\_test\_s)#Predicting the target for the given data

y\_pred

**Step 6: Accuracy/ Model Evaluation**

from sklearn.metrics import mean\_absolute\_percentage\_error,mean\_absolute\_error,r2\_score#checking for the accuracy of the prediction made in previous step

mean\_absolute\_percentage\_error(y\_test,y\_pred)

mean\_absolute\_error(y\_test,y\_pred)

r2\_score(y\_test,y\_pred)

**Note: Accuracies differ from model to model.**

**Step 7: Sample Future Prediction Example**

df\_new=df.sample(1)#taking a sample set from the dataset

df\_new#displaying sample set

X\_new=df\_new.drop(['Suburb','Address','SellerG','Date','Price'],axis=1)#defining X or independent variables of samole set

X\_new.shape

y\_pred\_new=model.predict(X\_new)#sample prediction

y\_pred\_new

**Classification Analysis:**

**Step 1: Define X and y**

y=df['Type']#'Type' is our Classification Target

X=df.drop(['Suburb','Address','SellerG','Date', 'Type'],axis=1)

**Dealing with Under-Sampling of Data**

**‘Type’**

from imblearn.under\_sampling import RandomUnderSampler

r=RandomUnderSampler(random\_state=2408)

**Before Oversampling**

X.shape, y.shape

y.value\_counts()

X.value\_counts()

**After Oversampling**

X, y = r.fit\_resample(X,y)

X.shape, y.shape

y.value\_counts()

X.value\_counts()

**Step 2: Splitting Data**

from sklearn.model\_selection import train\_test\_split

X\_train,X\_test,y\_train,y\_test=train\_test\_split(X,y,train\_size=0.7,stratify=y,random\_state=2408)

X\_train.shape,X\_test.shape,y\_train.shape,y\_test.shape

**Scaling the dataset**

df.describe()

from sklearn.preprocessing import StandardScaler

s=StandardScaler()

X\_train\_s=s.fit\_transform(X\_train)#Scaling train data

X\_test\_s=s.fit\_transform(X\_test)#Scaling test data

**Visualisation and Impact of Scaling**

* Reduced impact of outliers

X\_train\_s=pd.DataFrame(X\_train\_s,columns=X\_train.columns)

X\_test\_s=pd.DataFrame(X\_test\_s,columns=X\_test.columns)

fig, (ax1, ax2) = plt.subplots(ncols=2, figsize=(12, 5))

ax1.scatter(X\_train['YearBuilt'], X\_train[‘Price’],color='green')

ax1.set\_title("Before Scaling")

ax2.scatter (X\_train\_s['YearBuilt'], X\_train\_s['Price'],color="red")

ax2.set\_title("After Scaling")

plt.show()

**Note: Trying this code on different attributes will help you understand the impact of scaling**

**Step 3: Creating Model: 4 models of Classification**

**Model 1: Logistic Regression Model**

from sklearn.linear\_model import LogisticRegression

model=LogisticRegression()#Logistic Regression Model

**Model 2: KNN Model**

from sklearn.neighbors import KNeighborsClassifier

model=KNeighborsClassifier()#K-Nearest Neighbour Model

**Model 3: Decision Tree Model**

from sklearn.tree import DecisionTreeClassifier

model=DecisionTreeClassifier()#Decision tree Model

**Model 4: Support Vector Machine**

from sklearn.svm import SVC

model=SVC() #Support Vector Machine

**Note: Any one of the for models can be chosen. Accuracies vary according to the model that is chose.**

**Step 4: Training the Model**

model.fit(X\_train\_s,y\_train)#model that is used in the previous step is being trained in this step

**Step 5: Predicting Model**

y\_pred=model.predict(X\_test\_s)#Predicting the target for the given data

y\_pred

**Step 6: Accuracy/ Model Evaluation**

from sklearn.metrics import confusion\_matrix, classification\_report

#checking for the accuracy of the prediction made in previous step

#confusion matrix is the best way to represent the accuracy of a Classification Model. It represents True Positives to False Negatives

confusion\_matrix(y\_test,y\_pred)

print(classification\_report(y\_test,y\_pred))

**Note: Accuracies differ from model to model.**

**Step 7: Sample Future Prediction Example**

df\_new=df.sample(1)#taking a sample set from the dataset

df\_new#displaying sample set

X\_new=df\_new.drop(['Suburb','Address','SellerG','Date','Type'],axis=1)#defining X or independent variables of samole set

X\_new.shape

y\_pred\_new=model.predict(X\_new)#sample prediction

y\_pred\_new

Link of the Colab file: <https://colab.research.google.com/drive/1nZuh04PYrE9kxCzCDbyYsH-b3czkKgpD?usp=sharing>

**CHAPTER 5**

**RESULTS & CONCLUSION**

**Accuracies of Different Models:**

**I. Regression Analysis:**

**I.1. Linear Regression Model:** 61%

**I.2. KNN Model:** 99%

**I.3. Decision Tree Model:** 67%

**I.4. Random Forest Model:** 85%

Note: The variation in the frequencies is because of two main reasons. Firstly because it is a big data therefore it has a extreme limits or bounds and secondly, because the values are really close to each other or equal to each other making the Nearest Neighbour Model most accurate of all.

**II. Classification Analysis:**

**II.1. Logistic Regression Model:** 93%

**II.2. KNN Model:** 94%

**II.3. Decision Tree Model:** 92%

**II.4. Support Vector Machine:** 95%

Note: Random Forest Classifier is not involved in the project however it works with 96% Accuracy.

**CONCLUSION**

To recapitulate, it’s clear that the accuracy of KNN model in regression outweigh the accuracies of any other model so that should be the best model to choose whereas, on the other side, for classification any of the four models would give a faultless or precise result, ergo any one of them can be chosen.

This is how a Supervised Machine Learning model is to be prepared, trained, evaluated and deployed.

PDF: 

Colab File Link: <https://colab.research.google.com/drive/1nZuh04PYrE9kxCzCDbyYsH-b3czkKgpD?usp=sharing>

**REFERENCES**

1. github.com

2. Notes while classes

3. https://scikit-learn.org/stable/index.html

4. https://www.javatpoint.com/

5. https://www.geeksforgeeks.org/

6. https://en.wikipedia.org/

7. https://www.edureka.co/

8. https://www.seldon.io/

9. https://www.datarobot.com/

10. https://www.techtarget.com/searchbusinessanalytics/

11. https://bookdown.org/

12. https://www.analyticsvidhya.com/blog/

13. https://www.kdnuggets.com/

14. https://medium.com/

15. https://www.bankbazaar.com/

**THANK YOU!**