Business Report

Capstone Project

LI_BFSI_01+Life+Insurance+Sales

2nd Submission

Date: 25-Dec-2022

Created by Amit Jain

Table of Contents

List of Figure

1. Introduction of the business Problem

Introduction: This report explains the business requirements and provide the detailed solution based on the data provided for each problem statement. given in the assignment.

1.1 Defining Problem Statement:

"The dataset belongs to a leading life insurance company. The company wants to predict the bonus for its agents so that it may design appropriate engagement activity for their high performing agents and upskill programs for low performing agents are most important."

Dataset for Problem : **Sales.xlsx**

To understand the problem, Life insurance Company has given randomly collected sample of 4520 Customer records data in the sales.xlsx file, which have pattern of the Customer information about, their purchased Insurance Plans, their tenure with Insurance company, Sum insured and some more information about customer. Company has also given information about AgentBonus given to insurance company Agents who made customer purchase their plans.

Insurance company wants to analyze this sample data and Predict Bonus for their agents, based on past sample data , so that they can understand more about internal Agents, who bring sell to the Company. Company also want to know, if there are any low performing Agents, which requires any special training to increase growth or if they need any assistance. Company also want to build environment to encourage Agents, who are very good in selling plans, and motivate others by example of giving rewards to good performing Agents.

1.2 Need of the study/project:

It is very important for any company to know their Customers, at the same time its equally important to know their own employees, who serve end Customers. This is a very generic problem as well as requirement, specially in Insurance and Sales sectors to know their own employees, identifying good performing Agents/Sales person and low performing employees. So that they can know own capacity and can plan for the future growth. And based on this analysis, they can deploy their good performing Agents/Sales person in tough market and plan for good trainings to up scaling low performing employees.

1.3 Understanding business/social opportunity:

This kind of analysis for knowing own Agents/Sales person gives a good opportunity for any company to grow. They build a good healthy environment to encouraging good agents and upscaling and providing trainings assistance to not so good employees. It will also give a proper benchmark , to motivate other gents. By looking at the formula, management can understand, what parameters give more weightage for getting good compensation and good bonus amount. And it will remove partiality and bad judgement of the top level management.

2. Data Export :

We have been given Sample data from Insurance Company . We have performed following steps as part of 1st submissions :

1. Column name correction
2. Duplicate checks
3. NULL values treatments
4. Outlier treatments
5. Add new fields
6. Dropped un-necessary fields
7. Merging similar Categories of any column
8. Univariate analysis
9. Bivariate analysis
10. Multi variate analysis

We have exported all corrected data as part of our 1st assignments and again Imported that data for this report submission.

2.1 Visual inspection of data (rows, columns, descriptive details):

Import the data: Imported the data using Python notebooks and analyzed the effects of Education and Occupations over salary field.

This is how the data look like:

	Age	CustTenure	Channel	Occupation	EducationField	Gender	ExistingProdType	Designation	NumberOfPolicy	MaritalStatus	MonthlyIncome	Existing
0	22.0	4.0	Agent	Salaried	Graduate	Female	3	Manager	2.0	Single	20993	
1	11.0	2.0	Third Party Partner	Salaried	Graduate	Male	4	Manager	4.0	Divorced	20130	
2	26.0	4.0	Agent	Free Lancer	Post Graduate	Male	4	Executive	3.0	Unmarried	17090	
3	11.0	2.0	Third Party Partner	Salaried	Graduate	Female	3	Executive	3.0	Divorced	17909	
4	6.0	4.0	Agent	Small Business	Under Graduate	Male	3	Executive	4.0	Divorced	18468	

Data dictionary:

AgentBonus_per_policy => Average Bonus amount given to each agents in last month per Policy
Age =>Age of customer
CustTenure =>Tenure of customer in organization
Channel =>Channel through which acquisition of customer is done
Occupation =>Occupation of customer
EducationField =>Field of education of customer
Gender =>Gender of customer
ExistingProdType =>Existing product type of customer
Designation =>Designation of customer in their organization
NumberOfPolicy =>Total number of existing policy of a customer
MaritalStatus =>Marital status of customer
MonthlyIncome =>Gross monthly income of customer
ExistingPolicyTenure =>Max tenure in all existing policies of customer
SumAssured_per_policy => Average sum assured in all existing policies of customer per policy
Zone =>Customer belongs to which zone in India. Like East, West, North and South
PaymentMethod =>Frequency of payment selected by customer like Monthly, quarterly, half yearly and yearly

2.2 Understanding of attributes:

Data description:

	count	unique	top	freq	mean	std	min	25%	50%	75%	max
Age	4520.0	NaN	NaN	NaN	14.253761	8.945607	2.0	7.0	13.0	20.0	58.0
CustTenure	4520.0	NaN	NaN	NaN	13.910398	9.088495	1.0	7.0	12.0	19.0	57.0
Channel	4520	3	Agent	3194	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Occupation	4520	4	Salaried	2192	NaN	NaN	NaN	NaN	NaN	NaN	NaN
EducationField	4520	6	Graduate	1870	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Gender	4520	2	Male	2688	NaN	NaN	NaN	NaN	NaN	NaN	NaN
ExistingProdType	4520.0	NaN	NaN	NaN	3.688938	1.015769	1.0	3.0	4.0	4.0	6.0
Designation	4520	5	Executive	1662	NaN	NaN	NaN	NaN	NaN	NaN	NaN
NumberOfPolicy	4520.0	NaN	NaN	NaN	3.56969	1.449302	1.0	2.0	4.0	5.0	6.0
MaritalStatus	4520	4	Married	2268	NaN	NaN	NaN	NaN	NaN	NaN	NaN
MonthlyIncome	4520.0	NaN	NaN	NaN	22912.542699	4777.83029	16009.0	19850.0	21778.5	24701.0	38456.0
ExistingPolicyTenure	4520.0	NaN	NaN	NaN	4.797788	5.150644	1.0	2.0	3.0	6.0	52.0
Zone	4520	4	West	2566	NaN	NaN	NaN	NaN	NaN	NaN	NaN
PaymentMethod	4520	4	Half Yearly	2656	NaN	NaN	NaN	NaN	NaN	NaN	NaN
AgentBonus_Per_Policy	4520.0	NaN	NaN	NaN	1447.290306	1023.619093	341.666667	792.425	1117.333333	1737.25	8551.0
SumAssured_Per_Policy	4520.0	NaN	NaN	NaN	218463.40708	162075.165926	37268.0	116203.25	172236.5	262832.75	1400708.0

Insights:

1. Agent Bonus per policy: Bonus given to Agents, as well as Target variable. Minimum bonus given as 341 and Max is 8551
2. Age: Customers of all age group from 2 years to 58 years.
3. Customer Tenure: Many customers associated with Company from their Birth
4. Existing Prod Type : There are only 6 insurance products
5. Number of Policy: Customer can have multiple policies, from insurance company , for their family members or Self.
6. Monthly income Ranging between about 16K to 38K.

Data info:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4520 entries, 0 to 4519
Data columns (total 16 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   Age                    4520 non-null   float64
 1   CustTenure             4520 non-null   float64
 2   Channel                4520 non-null   object
 3   Occupation             4520 non-null   object
 4   EducationField         4520 non-null   object
 5   Gender                 4520 non-null   object
 6   ExistingProdType       4520 non-null   int64
 7   Designation            4520 non-null   object
 8   NumberOfPolicy         4520 non-null   float64
 9   MaritalStatus          4520 non-null   object
 10  MonthlyIncome          4520 non-null   int64
 11  ExistingPolicyTenure   4520 non-null   float64
 12  Zone                   4520 non-null   object
 13  PaymentMethod          4520 non-null   object
 14  AgentBonus_Per_Policy  4520 non-null   float64
 15  SumAssured_Per_Policy  4520 non-null   int64
dtypes: float64(5), int64(3), object(8)
memory usage: 565.1+ KB
```

List of the columns:

```
Index(['Age', 'CustTenure', 'Channel', 'Occupation', 'EducationField',
       'Gender', 'ExistingProdType', 'Designation', 'NumberOfPolicy',
       'MaritalStatus', 'MonthlyIncome', 'ExistingPolicyTenure', 'Zone',
       'PaymentMethod', 'AgentBonus_Per_Policy', 'SumAssured_Per_Policy'],
      dtype='object')
```

3. Data Preparation

3.1 Encoding:

In machine learning, we usually deal with datasets that contain multiple labels in one or more than one column. These labels can be in the form of words or numbers. To make the data understandable or in human-readable form, the training data is often labelled in words.

Label Encoding refers to converting the labels into a numeric form so as to convert them into the machine-readable form. Machine learning algorithms can then decide in a better way how those labels must be operated. It is an important pre-processing step for the structured dataset in supervised learning.

Limitation of label Encoding :

Label encoding converts the data in machine-readable form, but it assigns a unique number(starting from 0) to each class of data. This may lead to the generation of priority issues in the training of data sets. A label with a high value may be considered to have high priority than a label having a lower value.

Example:

An attribute having output classes Mexico, Paris, Dubai. On Label Encoding, this column lets Mexico is replaced with 0, Paris is replaced with 1, and Dubai is replaced with 2. With this, it can be interpreted that Dubai has high priority than Mexico and Paris while training the model, But actually, there is no such priority relation between these cities here.

After performing Encoding . this is how the data looks like for categorical fields :

```
Columns is : Channel
['Agent', 'Third Party Partner', 'Online']
Categories (3, object): ['Agent', 'Online', 'Third Party Partner']
[0 2 1]

Columns is : Occupation
['Salaried', 'Free Lancer', 'Small Business', 'Large Business']
Categories (4, object): ['Free Lancer', 'Large Business', 'Salaried', 'Small Business
']
[2 0 3 1]

Columns is : EducationField
['Graduate', 'Post Graduate', 'Under Graduate', 'Engineer', 'Diploma', 'MBA']
Categories (6, object): ['Diploma', 'Engineer', 'Graduate', 'MBA', 'Post Graduate', '
Under Graduate']
[2 4 5 1 0 3]

Columns is : Gender
['Female', 'Male']
Categories (2, object): ['Female', 'Male']
[0 1]

Columns is : Designation
['Manager', 'Executive', 'VP', 'AVP', 'Senior Manager']
Categories (5, object): ['AVP', 'Executive', 'Manager', 'Senior Manager', 'VP']
[2 1 4 0 3]
```

```
Columns is : MaritalStatus
['Single', 'Divorced', 'Unmarried', 'Married']
Categories (4, object): ['Divorced', 'Married', 'Single', 'Unmarried']
[2 0 3 1]

Columns is : Zone
['North', 'West', 'East', 'South']
Categories (4, object): ['East', 'North', 'South', 'West']
[1 3 0 2]

Columns is : PaymentMethod
['Half Yearly', 'Yearly', 'Quarterly', 'Monthly']
Categories (4, object): ['Half Yearly', 'Monthly', 'Quarterly', 'Yearly']
[0 3 2 1]
```

Sample looks like this :

	Age	CustTenure	Channel	Occupation	EducationField	Gender	ExistingProdType	Designation	NumberOfPolicy	MaritalStatus	MonthlyIncome	Existing
0	22.0	4.0	0	2	2	0	3	2	2.0	2	20993	
1	11.0	2.0	2	2	2	1	4	2	4.0	0	20130	
2	26.0	4.0	0	0	4	1	4	1	3.0	3	17090	
3	11.0	2.0	2	2	2	0	3	1	3.0	0	17909	
4	6.0	4.0	0	3	5	1	3	1	4.0	0	18468	

Data Info :

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4520 entries, 0 to 4519
Data columns (total 16 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Age                     4520 non-null   float64
 1   CustTenure              4520 non-null   float64
 2   Channel                 4520 non-null   int8
 3   Occupation              4520 non-null   int8
 4   EducationField          4520 non-null   int8
 5   Gender                  4520 non-null   int8
 6   ExistingProdType        4520 non-null   int64
 7   Designation             4520 non-null   int8
 8   NumberOfPolicy          4520 non-null   float64
 9   MaritalStatus           4520 non-null   int8
 10  MonthlyIncome           4520 non-null   int64
 11  ExistingPolicyTenure    4520 non-null   float64
 12  Zone                    4520 non-null   int8
 13  PaymentMethod           4520 non-null   int8
 14  AgentBonus_Per_Policy   4520 non-null   float64
 15  SumAssured_Per_Policy   4520 non-null   int64
```

3.2 Data Split: Split the data into train and test (70:30)

The train-test split is used to estimate the performance of machine learning algorithms that are applicable for prediction-based Algorithms/Applications. This method is a fast and easy procedure to perform such that we can compare our own machine learning model results to machine results. By default, the Test set is split into 30 % of actual data and the training set is split into 70% of the actual data.

We need to split a dataset into train and test sets to evaluate how well our machine learning model performs. The train set is used to fit the model, and the statistics of the train set are known. The second set is called the test data set, this set is solely used for predictions.

Dataset Splitting:

Scikit-learn alias sklearn is the most useful and robust library for machine learning in Python. The scikit-learn library provides us with the model_selection module in which we have the splitter function train_test_split().

Dependent variable : **AgentBonus_Per_Policy**

After performing train test split , We have got 4 parts :

```
Shape for X_train is  (3164, 15)
Shape for X_test is  (1356, 15)
Shape for y_train is  (3164, 1)
Shape for y_test is  (1356, 1)
```

```
X_train.head()
```

	Age	CustTenure	Channel	Occupation	EducationField	Gender	ExistingProdType	Designation	NumberOfPolicy	MaritalStatus	MonthlyIncome	Exis
2461	12.0	16.0	2	1	1	1	4	1	3.0	2	20742	
3681	31.0	15.0	0	3	5	1	4	2	5.0	1	23398	
1309	15.0	6.0	0	2	2	1	3	1	1.0	2	16232	
4254	5.0	16.0	1	1	1	1	4	2	2.0	1	23536	
1335	8.0	17.0	0	2	2	1	1	1	1.0	1	17269	

4. Model building and interpretation :

4.1 SKLearn Linear Model:

4.1.1 Definition:

Linear regression is defined as the process of determining the straight line that best fits a set of dispersed data points.

Linear regression is a simple and common type of predictive analysis. Linear regression attempts to model the relationship between two (or more) variables by fitting a straight line to the data. Put simply, linear regression attempts to predict the value of one variable, based on the value of another (or multiple other variables).

We can recall from high-school math that the equation for a linear relationship is: y = m(x) + b. In machine learning, m is often referred to as the weight of a relationship and b is referred to as the bias.

This relationship is referred to as a univariate linear regression because there is only a single independent variable. In many cases, our models won't actually be able to be predicted by a single independent variable. In these cases, there will be multiple independent variables influencing the dependent variable.

When working with scikit-linear learn's regression approach, we will encounter the following concepts:

- Best Fit - The straight line in a plot that minimizes the divergence between related dispersed data points
- Coefficient - Also known as a parameter, is the factor that is multiplied by a variable. A coefficient in linear regression represents changes in a Response Variable
- Coefficient of Determination - It is the correlation coefficient. In a regression, this term is used to define the precision or degree of fit
- Correlation - the measurable intensity and degree of association between two variables, often known as the 'degree of correlation.' The values range from -1.0 to 1.0
- Dependent Feature - A variable represented as y in the slope equation y=ax+b. Also referred to as an Output or a Response
- Estimated Regression Line - the straight line that best fits a set of randomly distributed data points
- Independent Feature - a variable represented by the letter x in the slope equation y=ax+b. Also referred to as an Input or a predictor
- Intercept - It is the point at where the slope intersects the Y-axis, indicated by the letter b in the slope equation y=ax+b
- Least Squares - a method for calculating the best fit to data by minimizing the sum of the squares of the discrepancies between observed and estimated values
- Mean - an average of a group of numbers; nevertheless, in linear regression, Mean is represented by a linear function

- Residual - the vertical distance between a data point and the regression line
- Regression - is an assessment of a variable's predicted change in relation to changes in other variables
- Regression Model - The optimum formula for approximating a regression
- Response Variables - This category covers both the Predicted Response (the value predicted by the regression) and the Actual Response (the actual value of the data point)
- Slope - the steepness of a regression line. The linear relationship between two variables may be defined using slope and intercept: y=ax+b

4.1.2 Action:

We have performed linear regression techniques from Sklearn LinearRegression library .

Following is co-efficient of Determination :

```
The coefficient for Age is 4.789430368850247
The coefficient for CustTenure is 4.759701081772676
The coefficient for Channel is -8.882960541752864
The coefficient for Occupation is 7.510211001225475
The coefficient for EducationField is -5.507046975671935
The coefficient for Gender is 2.1343057926728486
The coefficient for ExistingProdType is -15.273396744099568
The coefficient for Designation is 4.737994566790387
The coefficient for NumberOfPolicy is -159.7092491616902
The coefficient for MaritalStatus is 14.788341883378328
The coefficient for MonthlyIncome is 0.008683193446440198
The coefficient for ExistingPolicyTenure is 2.6087386706276594
The coefficient for Zone is -5.627466691687209
The coefficient for PaymentMethod is 1.6127186348940015
The coefficient for SumAssured_Per_Policy is 0.004765571407328226
```

4.1.3 Equation:

AgentBonus_per_policy = Age * 4.789430368850247 + CustTenure * 4.759701081772676 + Channel * -8.882960541752864 + Occupation * 7.510211001225475 + EducationField * -5.507046975671935 + Gender * 2.1343057926728486 + ExistingProdType * -15.273396744099568 + Designation * 4.737994566790387 + NumberOfPolicy * -159.7092491616902 + MaritalStatus * 14.788341883378328 + MonthlyIncome * 0.008683193446440198 + ExistingPolicyTenure * 2.6087386706276594 + Zone * -5.627466691687209 + PaymentMethod * 1.6127186348940015 + SumAssured_Per_Policy * 0.004765571407328226 + 671.7150861205953

The intercept for the model is 671.7150861205953
Intercept is the point on Y Axis , when all values of X are Zero. basically when we say, what should be the value of Y when all params are Zero

The coefficient of determination, denoted as R^2, tells you which amount of variation in y can be explained by the dependence on $\mathbf{x}$, using the particular regression model. A larger R^2 indicates a better fit and means that the model can better explain the variation of the output with different inputs.

The value R^2 = 1 corresponds to SSR = 0. That's the perfect fit, since the values of predicted and actual responses fit completely to each other.

R-Squared: R2 is a statistic that will give some information about the goodness of fit of a model. It ranges from 0 to 1. Exxample: if the value of R-squared is 0.745 so it explains 74% of variance is explained by the model.

for this Model , we have R-Squared as 0.896, which means, model is 89.6% of variance is explained by the model.

```
The coefficient of determination R^2 of the prediction on Train set 0.906776344656819
The coefficient of determination R^2 of the prediction on Test set 0.8964948926865645
```

4.1.4 Understanding Model:

One way to assess how well a regression model fits a dataset is to calculate the root mean square error, which is a metric that tells us the average distance between the predicted values from the model and the actual values in the dataset.

The lower the RMSE, the better a given model is able to "fit" a dataset.

The formula to find the root mean square error, often abbreviated RMSE, is as follows:

RMSE = Square root of (sum(Pi – Oi)**2 / n)

where:

Σ is a fancy symbol that means "sum"

Pi is the predicted value for the ith observation in the dataset

Oi is the observed value for the ith observation in the dataset

n is the sample size

This means that the RMSE represents the square root of the variance of the residuals.

suppose we want to build a regression model to predict the exam score of students and we want to find the best possible model among several potential models. Suppose we fit three different regression models and find their corresponding RMSE values:

RMSE of Model 1: 14.5

RMSE of Model 2: 16.7

RMSE of Model 3: 9.8

Model 3 has the lowest RMSE, which tells us that it's able to fit the dataset the best out of the three potential models.

```
The Root Mean Square Error (RMSE) of the model is for Train set is 317.48346002535106
The Root Mean Square Error (RMSE) of the model is for Train set is 316.6349977188605
```

This is how data points look like for predicted variables:

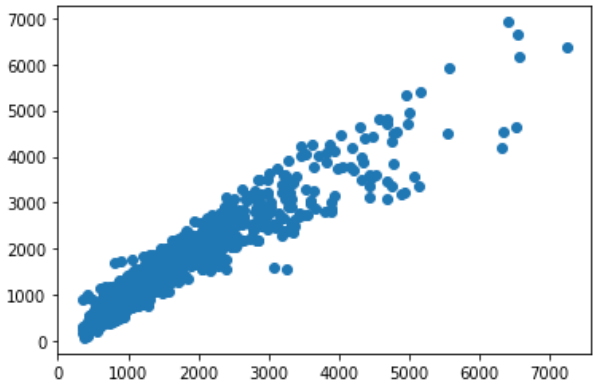


Figure 1 Scatter plot for Prediction on Test data

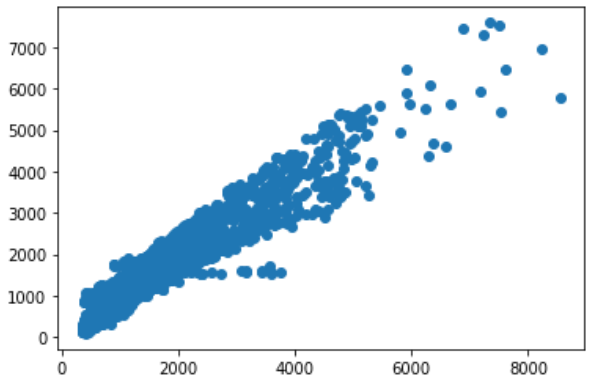


Figure 2 Scatter plot for Prediction on Train data

4.1.5 Performance parameters:

Train data matrix:

```
The MAE is: 220.32100694973653
The MSE is: 100795.7473896687
The MAPE is 0.1707241803718864
The EVS is 0.9067763446568196
The RMSE 316.6349977188605
```

Test data matrix:

```
The MAE for Test data is: 222.5194900195983
The MSE for Test data is: 100257.72178042283
The MAPE for Test data is 0.17126532112977724
The EVS for Test data is 0.8964952272518002
The RMSE 317.48346002535106
```

4.1.6 Feature importance

Feature importance refers to a class of techniques for assigning scores to input features to a predictive model that indicates the relative importance of each feature when making a prediction.

We fit a LinearRegression model on the regression dataset and retrieve the *coeff_* property that contains the coefficients found for each input variable. These coefficients can provide the basis for a crude feature importance score. This assumes that the input variables have the same scale or have been scaled prior to fitting a model.

In regression analysis, the magnitude of coefficients is not necessarily 100% related to their importance. The most common criteria to determine the importance of independent variables in regression analysis are **p-values**.

Small p-values imply high levels of importance, whereas high p-values mean that a variable is not statistically significant.

We can only use the magnitude of coefficients as a measure for feature importance when our model is penalizing variables. **sklearn does not report p-values though.**

4.2 Building Stats Model:

4.2.1 Definition:

Linear regression statsmodel is the model that helps us to predict and is used for fitting up the scenario where one parameter is directly dependent on the other parameter. Here, we have one variable that is dependent and the other one which is independent. Depending on the change in the value of the independent parameter, we need to predict the change in the dependent variable.

Ordinary least squares (OLS) regression is a statistical method of analysis that estimates the relationship between one or more independent variables and a dependent variable; the method estimates the relationship by minimizing the sum of the squares in the difference between the observed and predicted values of the dependent variable configured as a straight line.

4.2.2 Action:

OLS model works on only train and Test data set, it does not need separation of Dependent and Independent variables . We just need to specify during model building , which variable is Dependent . but Sample data don't need a separation in it.

So, we have merged X and Y variables and built separate Train and Test data set .

Field names for both train and test data set are:

```
Index(['Age', 'CustTenure', 'Channel', 'Occupation', 'EducationField',
       'Gender', 'ExistingProdType', 'Designation', 'NumberOfPolicy',
       'MaritalStatus', 'MonthlyIncome', 'ExistingPolicyTenure', 'Zone',
       'PaymentMethod', 'SumAssured_Per_Policy', 'AgentBonus_Per_Policy'])
```

Data Info :

```
#   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   Age                    3164 non-null   float64
 1   CustTenure             3164 non-null   float64
 2   Channel                3164 non-null   int8
 3   Occupation             3164 non-null   int8
 4   EducationField         3164 non-null   int8
 5   Gender                 3164 non-null   int8
 6   ExistingProdType       3164 non-null   int64
 7   Designation            3164 non-null   int8
 8   NumberOfPolicy         3164 non-null   float64
 9   MaritalStatus          3164 non-null   int8
 10  MonthlyIncome          3164 non-null   int64
 11  ExistingPolicyTenure   3164 non-null   float64
 12  Zone                   3164 non-null   int8
 13  PaymentMethod          3164 non-null   int8
 14  SumAssured_Per_Policy  3164 non-null   int64
 15  AgentBonus_Per_Policy  3164 non-null   float64
```

OLS Regression Results

Dep. Variable:	AgentBonus_Per_Policy	R-squared:	0.907
Model:	OLS	Adj. R-squared:	0.906
Method:	Least Squares	F-statistic:	2041.
Date:	Thu, 22 Dec 2022	Prob (F-statistic):	0.00
Time:	21:46:41	Log-Likelihood:	-22716.
No. Observations:	3164	AIC:	4.546e+04
Df Residuals:	3148	BIC:	4.556e+04
Df Model:	15		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	671.7151	46.634	14.404	0.000	580.280	763.151
Age	4.7894	0.712	6.729	0.000	3.394	6.185
CustTenure	4.7597	0.726	6.559	0.000	3.337	6.183
Channel	-8.8830	7.170	-1.239	0.215	-22.941	5.175
Occupation	7.5102	10.200	0.736	0.462	-12.489	27.509
EducationField	-5.5070	3.704	-1.487	0.137	-12.770	1.756
Gender	2.1343	11.518	0.185	0.853	-20.449	24.718
ExistingProdType	-15.2734	6.635	-2.302	0.021	-28.283	-2.263
Designation	4.7380	6.455	0.734	0.463	-7.918	17.394
NumberOfPolicy	-159.7092	5.825	-27.418	0.000	-171.130	-148.288
MaritalStatus	14.7883	7.443	1.987	0.047	0.195	29.382
MonthlyIncome	0.0087	0.002	5.480	0.000	0.006	0.012
ExistingPolicyTenure	2.6087	1.260	2.070	0.039	0.138	5.080
Zone	-5.6275	5.606	-1.004	0.316	-16.619	5.365
PaymentMethod	1.6127	4.787	0.337	0.736	-7.772	10.998
SumAssured_Per_Policy	0.0048	5.24e-05	91.008	0.000	0.005	0.005

Omnibus:	1340.129	Durbin-Watson:	1.952
Prob(Omnibus):	0.000	Jarque-Bera (JB):	11352.537
Skew:	1.794	Prob(JB):	0.00
Kurtosis:	11.558	Cond. No.	2.29e+06

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 2.29e+06. This might indicate that there are
strong multicollinearity or other numerical problems.

After building OLS Model , this is the co-efficient of determinations:

Intercept	671.715086
Age	4.789430
CustTenure	4.759701
Channel	-8.882961
Occupation	7.510211
EducationField	-5.507047
Gender	2.134306
ExistingProdType	-15.273397
Designation	4.737995
NumberOfPolicy	-159.709249
MaritalStatus	14.788342
MonthlyIncome	0.008683
ExistingPolicyTenure	2.608739
Zone	-5.627467
PaymentMethod	1.612719
SumAssured_Per_Policy	0.004766

4.2.3 Performance parameters:

Train data matrix:

```
The MAE is: 220.32100694970435
The MSE is: 100795.74738966869
The MAPE is 0.17072418037186507
The EVS is 0.9067763446568196
The RMSE 316.63499771886285
```

Test data matrix:

```
The MAE for Test data is: 222.5194900195671
The MSE for Test data is: 100257.7217804241
The MAPE for Test data is 0.17126532112975362
The EVS for Test data is 0.8964952272517993
The RMSE 317.48346002535055
```

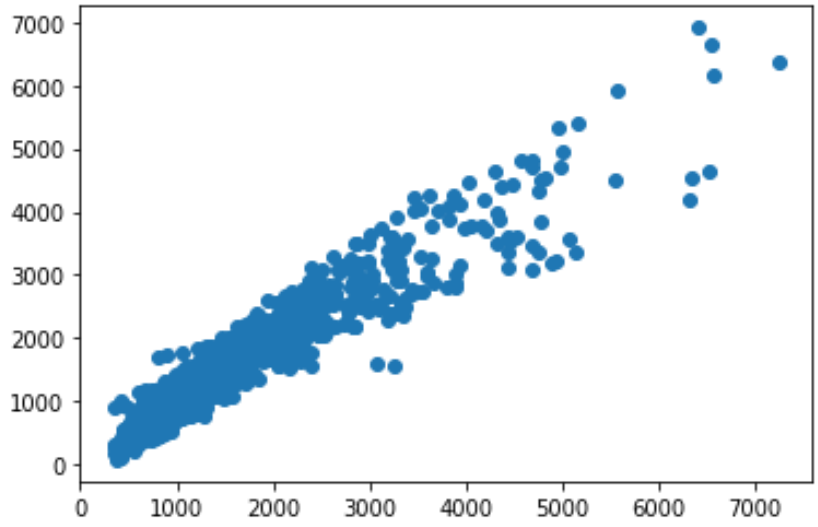


Figure 3Scatter Plot for Predicted values on Test data set

4.2.4 Feature Importance:

In regression analysis, the magnitude of coefficients is not necessarily 100% related to their importance. The most common criteria to determine the importance of independent variables in regression analysis are **p-values**.

Small p-values imply high levels of importance, whereas high p-values mean that a variable is not statistically significant.

We can only use the magnitude of coefficients as a measure for feature importance when our model is penalizing variables. **sklearn does not report p-values though.** But Stats model does give **p-values** .

From Stats model summary , we can see that there are high P value for many of the fields from our data set.

	coef	P>|t|
Intercept	671.7151	0
Age	4.7894	0
CustTenure	4.7597	0
Channel	-8.883	0.215
Occupation	7.5102	0.462
EducationField	-5.507	0.137
Gender	2.1343	0.853
ExistingProdType	-15.2734	0.021
Designation	4.738	0.463
NumberOfPolicy -	159.7092	0
MaritalStatus	14.7883	0.047
MonthlyIncome	0.0087	0
ExistingPolicyTenure	2.6087	0.039
Zone	-5.6275	0.316
PaymentMethod	1.6127	0.736
SumAssured_Per_Policy	0.0048	0

So we can consider High Importance features are :

Age , CustomerTenure, NumberofPolicy, MonthlyIncome and SumAssured_Per_Policy

And less affecting features are :

PaymentMethod , Gender, Occupation, Designation etc.

4.3 Building Multiple Stats Model to eliminate non affecting fields :

4.3.1 Stats Model for fields with lower value of VIF :

Calculate VIF for each Field and based on VIF value we decide, which column can be eliminated.

VIF (variance inflation factor) is used for checking multicollinearity in regression Model. its Values can range between 1 to Infinite.

4.3.1.1 How to interpret a given VIF value?:

Consider the following linear regression model: **$Y = \beta 0 + \beta 1 \times X1 + \beta 2 \times X2 + \beta 3 \times X3 + \varepsilon$**

For each of the independent variables X1, X2 and X3 we can calculate the variance inflation factor (VIF) in order to determine if we have a multicollinearity problem.

Here's the formula for calculating the VIF for X1: **VIF1 = 1 / (1 - R**2)**

VIF (variance inflating factor) formula for the first variable in the model.
R2 in this formula is the coefficient of determination from the linear regression model which has:
X1 as dependent variable

X2 and X3 as independent variables

In other words, R2 comes from the following linear regression model:

$X1 = \beta0 + \beta1 \times X2 + \beta2 \times X3 + \varepsilon$

And because R2 is a number between 0 and 1:
When R2 is close to 1 (i.e. X2 and X3 are highly predictive of X1): the VIF will be very large
When R2 is close to 0 (i.e. X2 and X3 are not related to X1): the VIF will be close to 1
Therefore the range of VIF is between 1 and infinity.

4.3.1.2 Action:

We have checked VIF for each field, and taken threshold to 5. Fields with more than value of 5, can be consider to eliminate that field . We have done this exercise in a For Loop,

Out of 16 fields, we have consider only these 10 fields, which have VIF value less than 5 :

```
Index(['CustTenure', 'Channel', 'EducationField', 'Gender', 'Designation',
       'MaritalStatus', 'ExistingPolicyTenure', 'Zone', 'PaymentMethod',
       'SumAssured_Per_Policy'],
```

VIF Values for all of above fields are :

```
VIF for  CustTenure --> 3.7649121302806456
VIF for  Channel --> 1.351814184733024
VIF for  EducationField --> 3.013828674898219
VIF for  Gender --> 2.23880673655966
VIF for  Designation --> 3.6622884417664836
VIF for  MaritalStatus --> 2.9365604234253677
VIF for  ExistingPolicyTenure --> 2.182083152055937
VIF for  Zone --> 4.306208853382339
VIF for  PaymentMethod --> 1.5613624945213813
VIF for  SumAssured_Per_Policy --> 2.890013748992411
```

Co-efficient of determination :

```
Intercept                97.578366
CustTenure                2.490252
Channel                  -5.525502
EducationField           -3.716762
Gender                   12.087670
Designation               9.399802
MaritalStatus            16.199391
ExistingPolicyTenure      2.282474
Zone                      1.504290
PaymentMethod            -5.626124
SumAssured_Per_Policy     0.005840
dtype: float64
```

OLS Regression Results

Dep. Variable:	AgentBonus_Per_Policy	R-squared:	0.884
Model:	OLS	Adj. R-squared:	0.883
Method:	Least Squares	F-statistic:	2397.
Date:	Thu, 22 Dec 2022	Prob (F-statistic):	0.00
Time:	21:46:42	Log-Likelihood:	-23065.
No. Observations:	3164	AIC:	4.615e+04
Df Residuals:	3153	BIC:	4.622e+04
Df Model:	10		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	97.5784	26.937	3.622	0.000	44.763	150.394
CustTenure	2.4903	0.760	3.277	0.001	1.000	3.980
Channel	-5.5255	7.992	-0.691	0.489	-21.196	10.145
EducationField	-3.7168	3.618	-1.027	0.304	-10.810	3.376
Gender	12.0877	12.840	0.941	0.347	-13.089	37.264
Designation	9.3998	6.596	1.425	0.154	-3.533	22.332
MaritalStatus	16.1994	8.264	1.960	0.050	-0.003	32.402
ExistingPolicyTenure	2.2825	1.309	1.743	0.081	-0.285	4.850
Zone	1.5043	6.246	0.241	0.810	-10.742	13.751
PaymentMethod	-5.6261	4.639	-1.213	0.225	-14.721	3.469
SumAssured_Per_Policy	0.0058	3.95e-05	147.880	0.000	0.006	0.006

Omnibus:	1424.436	Durbin-Watson:	1.955
Prob(Omnibus):	0.000	Jarque-Bera (JB):	11772.840
Skew:	1.945	Prob(JB):	0.00
Kurtosis:	11.612	Cond. No.	1.21e+06

4.3.1.3 Performance parameters:

Train data matrix:

```
The MAE is: 238.54187545716903
The MSE is: 125701.42525909047
The MAPE is 0.17557448996078645
The EVS is 0.8837416592666577
The RMSE 353.56277958413546
```

Test data matrix:

```
The MAE for Test data is: 244.14186309381338
The MSE for Test data is: 125006.63910726018
The MAPE for Test data is 0.1796778764535846
The EVS for Test data is 0.8709648906909357
The RMSE 354.54396801961036
```

4.3.1.4 Stats Model for fields with Lower P value and remove other fields:

Based on the OLS Summary, form above model, we have come to this conclusion that,, there are many fields, with High P -value, the Higher the P Value, Lower the chance it will affect the Predictions. This is the list of P value for all the fields:

	coef	P>|t|
Intercept	97.5784	0
CustTenure	2.4903	0.001
Channel	-5.5255	0.489
EducationField	-3.7168	0.304
Gender	12.0877	0.347
Designation	9.3998	0.154
MaritalStatus	16.1994	0.05
ExistingPolicyTenure	2.2825	0.081
Zone	1.5043	0.81
PaymentMethod	-5.6261	0.225
SumAssured_Per_Policy	0.0058	0

4.3.1.5 Action:

We have removed fields one by one and re-generated OLS model and checked P values for fields again and again, until each field importance came below 0.05.

We started with these fields:

1. Model Build by removing field name "**Zone**" : Since Zone has Highest P-Value , so we dropped this field and Built our Model, on top of Step 4.3.1 , where we eliminated fields with higher VIF.
   After building Model without field "Zone", We still saw that there were many fields with High P-Value .

2. Model Build by removing field name "**Channel**": Since Channel , has Highest P-Value , so we dropped this field and Built our Model on top of above model. We still saw that there were many fields with High P-Value .

3. Model Build by removing field name "**Gender**": Since Gender, has Highest P-Value , so we dropped this field and Built our Model on top of above model. We still saw that there were many fields with High P-Value .

4. Model Build by removing field name "**EducationField**": Since **EducationField**, has Highest P-Value , so we dropped this field and Built our Model on top of above model. We still saw that there were many fields with High P-Value .

5. Model Build by removing field name "**PaymentMethod**": Since **PaymentMethod** , has Highest P-Value , so we dropped this field and Built our Model on top of above model. We still saw that there were many fields with High P-Value .

6. Model Build by removing field name "**Designation**": Since **Designation**, has Highest P-Value , so we dropped this field and Built our Model on top of above model. We still saw that there were many fields with High P-Value .

7. Model Build by removing field name "**MaritalStatus**": Since **MaritalStatus**, has Highest P-Value , so we dropped this field and Built our Model on top of above model. We still saw that there were many fields with High P-Value .

   After dropping **MaritalStatus ,** we Built our Final Stats Model with these features:
   No Field have VIF value more than 5
   No field have higher p-Value than 0.05

   And we are left with only these 3 fields, which are most important feature to determine the targeted values:

   CustTenure
   ExistingPolicyTenure
   SumAssured_per_policy

```
                            OLS Regression Results
==============================================================================
Dep. Variable:     AgentBonus_Per_Policy   R-squared:                       0.883
Model:                               OLS   Adj. R-squared:                  0.883
Method:                    Least Squares   F-statistic:                     7980.
Date:                   Thu, 22 Dec 2022   Prob (F-statistic):               0.00
Time:                           21:46:44   Log-Likelihood:                -23070.
No. Observations:                   3164   AIC:                         4.615e+04
Df Residuals:                       3160   BIC:                         4.617e+04
Df Model:                              3
Covariance Type:               nonrobust
===========================================================================================
                              coef    std err          t      P>|t|      [0.025      0.975]
-------------------------------------------------------------------------------------------
Intercept               121.5852     13.382      9.085      0.000      95.346     147.824
CustTenure                2.5282      0.753      3.356      0.001       1.051       4.005
ExistingPolicyTenure      2.2797      1.306      1.745      0.081      -0.282       4.841
SumAssured_Per_Policy     0.0058   3.93e-05    148.868      0.000       0.006       0.006
==============================================================================
Omnibus:                     1424.859   Durbin-Watson:                   1.950
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            11780.897
Skew:                           1.946   Prob(JB):                         0.00
Kurtosis:                      11.615   Cond. No.                     5.84e+05
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 5.84e+05. This might indicate that there are
strong multicollinearity or other numerical problems.
```

4.3.1.6 Performance parameters:

Train data matrix:

```
The MAE is: 238.6871961437201
The MSE is: 126072.974963021
The MAPE is 0.17554253326507824
The EVS is 0.8833980215394808
The RMSE 352.60212402026525
```

Test data matrix:

```
The MAE for Test data is: 243.29840734002366
The MSE for Test data is: 124328.25786360275
The MAPE for Test data is 0.1787555654744601
The EVS for Test data is 0.8716758974028127
The RMSE 355.0675639410347
```

4.3.1.7 Equation and interpretation:

We can use below formula for determining the targeted value:

AgentBonus_per_policy = (121.59) + (2.53) * CustTenure + (2.28) * ExistingPolicyTenure + (0.01) * SumAssured_Per_Policy

We can see from the equation, if all of the Independent parameters are Zero, then also Agent will receive a Bonus of Rs 121.59 , which is very good.
Also there is clear that 1% of total amount of SumAssured goes into Agent Bonus, because co-efficient is 0.01

4.4 Decision Tree Model

4.4.1 Definition:

Decision Tree is a decision-making tool that uses a flowchart-like tree structure or is a model of decisions and all of their possible results, including outcomes, input costs, and utility. Decision-tree algorithm falls under the category of supervised learning algorithms. It works for both continuous as well as categorical output variables.

The branches/edges represent the result of the node and the nodes have either:
Conditions [Decision Nodes]
Result [End Nodes]

The branches/edges represent the truth/falsity of the statement and take makes a decision based on that in the example below which shows a decision tree that evaluates the smallest of three numbers:

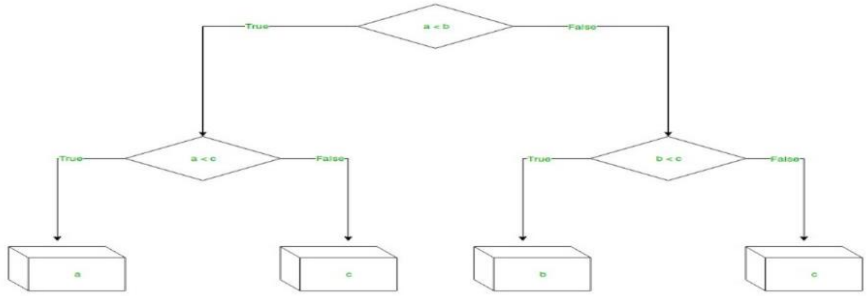


Figure 4 Example for Decision tree

Decision Tree Regression:
Decision tree regression observes features of an object and trains a model in the structure of a tree to predict data in the future to produce meaningful continuous output. Continuous output means that the output/result is not discrete, i.e., it is not represented just by a discrete, known set of numbers or values.
Discrete output example: A weather prediction model that predicts whether or not there'll be rain on a particular day.
Continuous output example: A profit prediction model that states the probable profit that can be generated from the sale of a product.

4.4.2 Action:

Again, decision tree works on Separate group of dependent variable and independent variables of train and test data. So we will use our Old sub groups of train and test split data set:

After performing train test split , we have got 4 parts :

```
Shape for X_train is  (3164, 15)
Shape for X_test is  (1356, 15)
Shape for y_train is  (3164, 1)
Shape for y_test is  (1356, 1)
```

We have used Sklearn DecisionTreeRegressor library for our model building and created Model .

4.4.3 Feature Importance :

After building decision tree Model, this is the list of features with their importance weightage

```
SumAssured_Per_Policy    0.879796
NumberOfPolicy           0.041685
MonthlyIncome            0.023623
Age                      0.018812
CustTenure               0.014237
ExistingPolicyTenure     0.008178
Designation              0.002733
MaritalStatus            0.001986
Occupation               0.001936
ExistingProdType         0.001746
EducationField           0.001732
Zone                     0.000928
PaymentMethod            0.000918
Gender                   0.000906
Channel                  0.000786
```

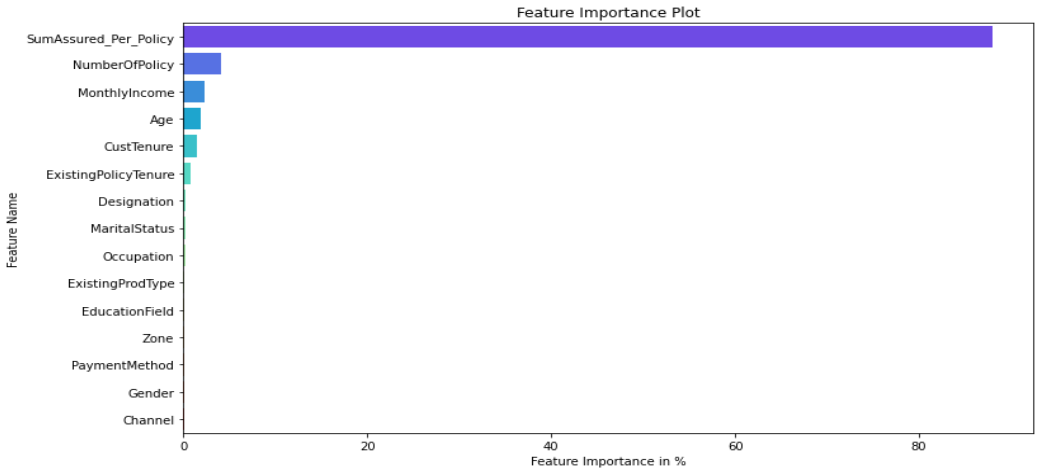


Figure 5 Decision Tree Feature Importance

4.4.4 Performance parameters:

Train data matrix:

```
The MAE is: 0.0
The MSE is: 0.0
The MAPE is 0.0
The EVS is 1.0
The RMSE is:0.00
```

Test data matrix:

```
The MAE for Test data is: 239.35693215339234
The MSE for Test data is: 136407.9355465421
The MAPE for Test data is 0.17331057475482273
The EVS for Test data is 0.8592843399780032
The RMSE is:369.33
```

Decision Tree model always give a clear separation on Train data set, that's why it gives value of 0 for all the performance parameters, When we checked for test data, then actual errors comes, and RMSE for DTree is 369.33

4.5 Random Forest Regressor Model

4.5.1 Definition :

Every decision tree has high variance, but when we combine all of them together in parallel then the resultant variance is low as each decision tree gets perfectly trained on that particular sample data, and hence the output doesn't depend on one decision tree but on multiple decision trees. In the case of a classification problem, the final output is taken by using the majority voting classifier. In the case of a regression problem, the final output is the mean of all the outputs. This part is called **Aggregation**.

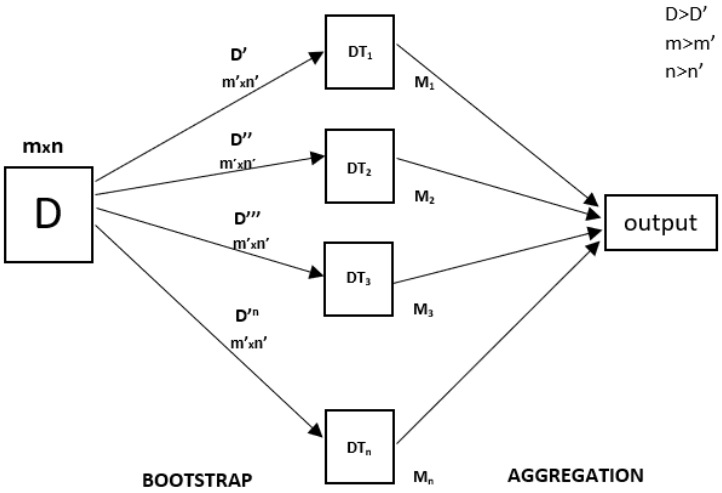


Figure 6 Example for Random Forest

Random Forest is an ensemble technique capable of performing both regression and classification tasks with the use of multiple decision trees and a technique called Bootstrap and Aggregation, commonly known as **bagging**. The basic idea behind this is to combine multiple decision trees in determining the final output rather than relying on individual decision trees.
Random Forest has multiple decision trees as base learning models. We randomly perform row sampling and feature sampling from the dataset forming sample datasets for every model. This part is called Bootstrap.
We need to approach the Random Forest regression technique like any other machine learning technique

4.5.2 Action:

Random forest is similar to decision tree, which works on Separate group of dependent variable and independent variables of train and test data. So we will use our Old sub groups of train and test split data set:

After performing train test split , we have got 4 parts :

```
Shape for X_train is  (3164, 15)
Shape for X_test is  (1356, 15)
Shape for y_train is  (3164, 1)
Shape for y_test is  (1356, 1)
```

We have used Sklearn RandomForestRegressor library for our model building and created Model .

4.5.3 Feature Importance :

After building Renadom Forest Model, this is the list of features with their importance weightage

```
SumAssured_Per_Policy    0.891987
NumberOfPolicy           0.038844
Age                      0.016303
MonthlyIncome            0.015429
CustTenure               0.014407
ExistingPolicyTenure     0.008537
EducationField           0.002414
Designation              0.002192
ExistingProdType         0.001839
MaritalStatus            0.001773
Occupation               0.001504
Gender                   0.001254
Channel                  0.001195
PaymentMethod            0.001186
Zone                     0.001135
```

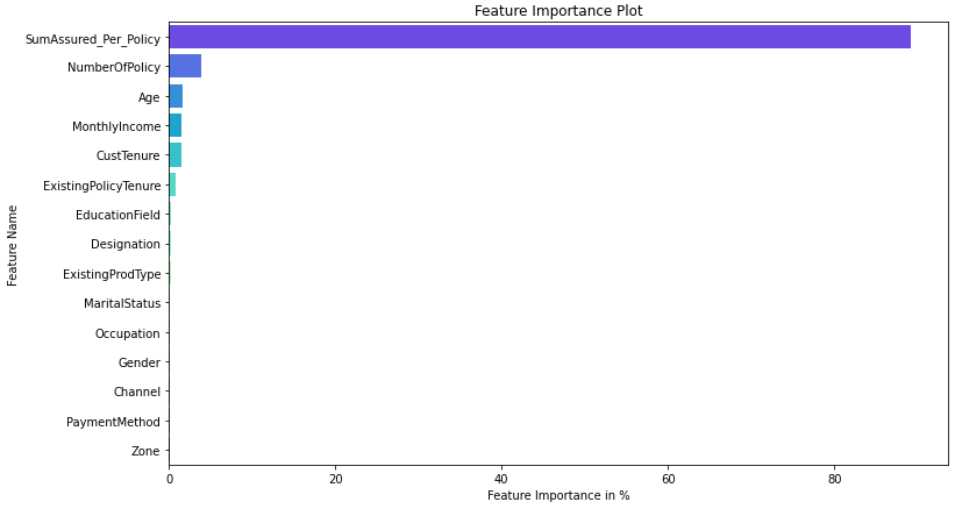


Figure 7Random Forest Feature Importance

4.5.4 Performance parameters:

Train data matrix:

```
The MAE is: 65.69539559629162
The MSE is: 10394.932540860822
The MAPE is 0.04768645807947594
The EVS is 0.9903860465401678
The RMSE is:101.96
```

Test data matrix:

```
The MAE for Test data is: 178.33922976073418
The MSE for Test data is: 73391.11692993417
The MAPE for Test data is 0.1282120366478216
The EVS for Test data is 0.9242322882676777
The RMSE is:270.91
```

4.6 Artificial Neural Network Regressor

The purpose of using Artificial Neural Networks for Regression over Linear Regression is that the linear regression can only learn the linear relationship between the features and target and therefore cannot learn the complex non-linear relationship. In order to learn the complex non-linear relationship between the features and target, we are in need of other techniques. One of those techniques is to use Artificial Neural Networks. Artificial Neural Networks have the ability to learn the complex relationship between the features and target due to the presence of activation function in each layer. Let's look at what are Artificial Neural Networks and how do they work.

4.6.1 Definition:

Artificial Neural Networks are one of the deep learning algorithms that simulate the workings of neurons in the human brain. There are many types of Artificial Neural Networks, Vanilla Neural Networks, Recurrent Neural Networks, and Convolutional Neural Networks. The Vanilla Neural Networks have the ability to handle structured data only, whereas the Recurrent Neural Networks and Convolutional Neural Networks have the ability to handle unstructured data very well.

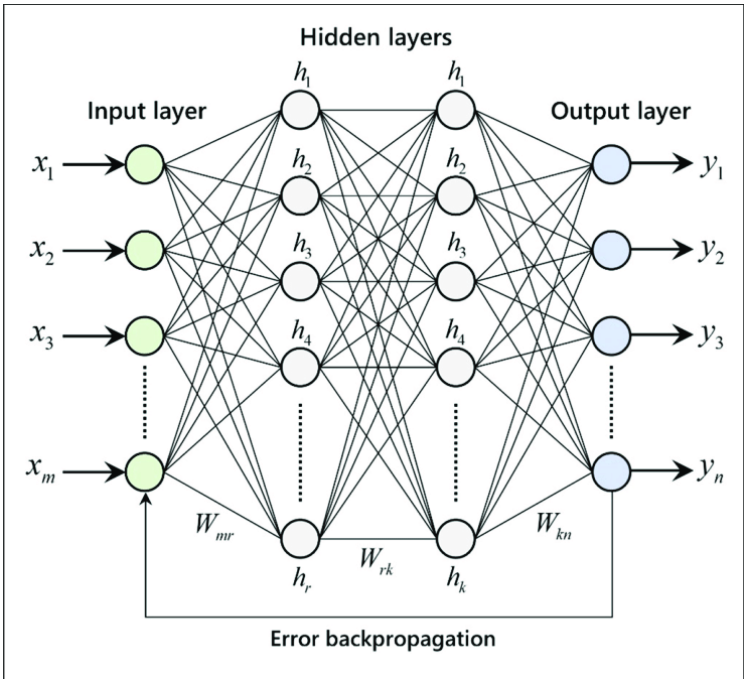


Figure 8 Example for ANN Model

The Artificial Neural Networks consists of the Input layer, Hidden layers, Output layer. The hidden layer can be more than one in number. Each layer consists of n number of neurons. Each layer will be having an Activation Function associated with each of the neurons. The activation function is the function that is responsible for introducing non-linearity in the relationship. In our case, the output layer must contain a linear activation function. Each layer can also have regularizers associated with it. Regularizers are responsible for preventing overfitting.

The functionality of ANN can be explained in below 5 simple steps:

1. Read the input data
2. Produce the predictive model (A mathematical function)
3. Measure the error in the predictive model
4. Inform and implement necessary corrections to the model repeatedly until a model with least error is found
5. Use this model for predicting the unknown

4.6.2 Action:

We have used Grid search to find the best parameters used in our prediction model. We have used following params for grid search:

```
param_grid = {    'hidden_layer_sizes': [50,100],
   'activation':['logistic','relu'],
   'max_iter': [250,],
   'solver': ['adam','sgd'],
   'tol': [0.1,0.01], }
```

We want our Model to check between 50 and 100 hidden layers, the more number of hidden layers you apply, more time it will take to execute.
The activation function generates an output based on input signals. Logistic/Segmoid are same

After using grid search these are the Parameters, we have received :

```
{'activation': 'logistic',
 'hidden_layer_sizes': 100,
 'max_iter': 250,
 'solver': 'sgd',
 'tol': 0.1}
```

sigmoid/logistic activation function is one of the most widely used functions:

- It is commonly used for models where we have to predict the probability as an output. Since probability of anything exists only between the range of 0 and 1, sigmoid is the right choice because of its range.
- The function is differentiable and provides a smooth gradient, i.e., preventing jumps in output values. This is represented by an S-shape of the sigmoid activation function.

4.6.3 Feature Importance :

We cannot get the values of list of best parameters used for prediction Model , because it all comes under Hidden layers.

4.6.4 Performance parameters:

Train data matrix:

```
The MAE is: 4972.297786270588
The MSE is: 25833654.247346397
The MAPE is 5.255688936394857
The EVS is -0.09295335197465215
The RMSE is:5082.68
```

Test data matrix:

```
The MAE for Test data is: 4990.048298012509
The MSE for Test data is: 25943347.914660286
The MAPE for Test data is 5.291987651047779
The EVS for Test data is -0.09990145029430497
The RMSE is:5093.46
```

5. Model Tuning and business implication

There are many ways we can Tune our Model :

1. We can use bagging/Boosting regressors
2. We can Tune and search for best Hyper parameters.
3. We can KFold our data and do testing

I have used above methods and built following models :

5.1 Bagging Regressor (Random Forest should be applied for Bagging)

5.1.1 Definition :

Bagging, also known as Bootstrap aggregating, is an ensemble learning technique that helps to improve the performance and accuracy of machine learning algorithms. It is used to deal with bias-variance trade-offs and reduces the variance of a prediction model. Bagging avoids overfitting of data and is used for both regression and classification models, specifically for decision tree algorithms.

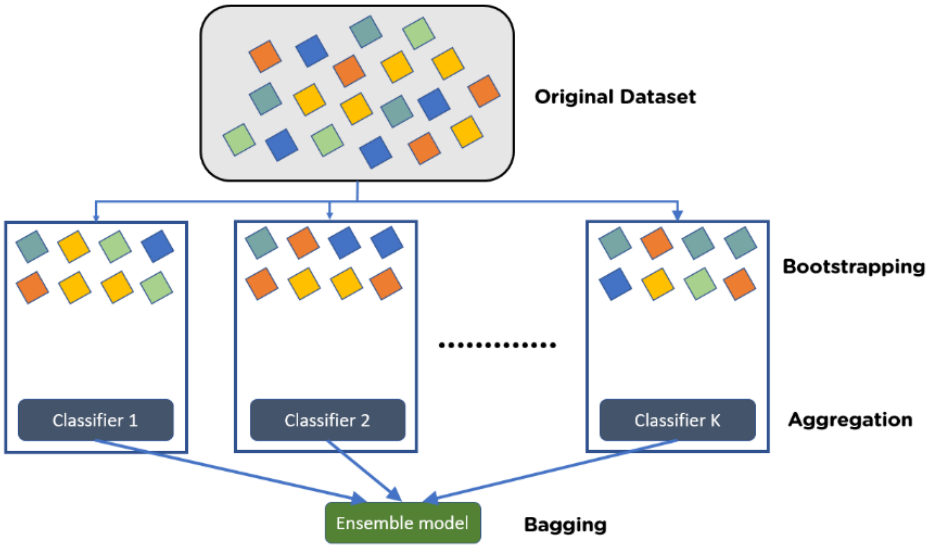


Figure 9 Explain Bagging

Bootstrapping is the method of randomly creating samples of data out of a population with replacement to estimate a population parameter.

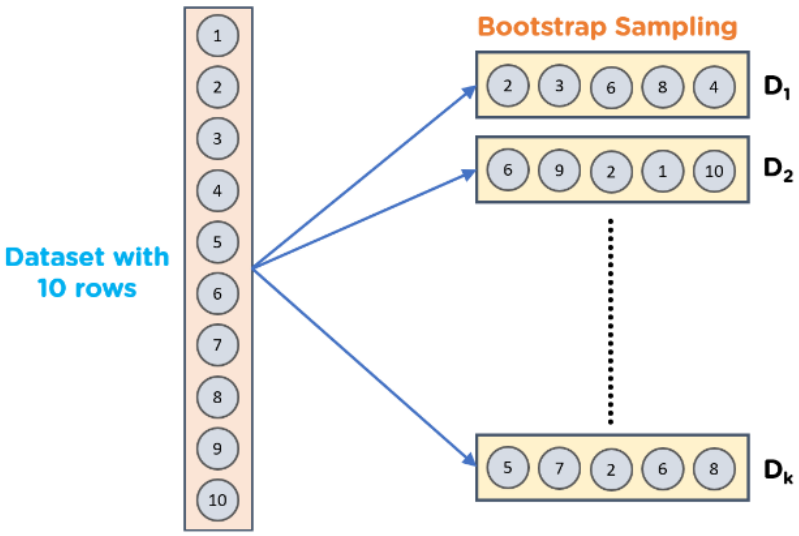


Figure 10 Explain Bootstrapping

Steps to Perform Bagging:

- Consider there are n observations and m features in the training set. You need to select a random sample from the training dataset without replacement
- A subset of m features is chosen randomly to create a model using sample observations
- The feature offering the best split out of the lot is used to split the nodes
- The tree is grown, so you have the best root nodes
- The above steps are repeated n times. It aggregates the output of individual decision trees to give the best prediction

5.1.2 Action:

I have used Random Forest regressor for base_estimator and then used base_estimator parameters of Random Forest into Bagging Algorithm.

I have used Sklearn BaggingRegressor library for our model building.

5.1.3 Feature Importance :

Bagging don't give List of best features. 'BaggingRegressor' object has no attribute 'feature_importances_'

So we cannot say, which Attribute of Sample data set is more contributing in prediction model

5.1.4 Performance parameters:

Train data matrix:

```
The MAE is: 106.47556038137378
The MSE is: 24820.70796807343
The MAPE is 0.07690297993131891
The EVS is 0.9770456925867458
The RMSE is:157.55
```

Test data matrix:

```
The MAE for Test data is: 180.50244943952802
The MSE for Test data is: 72519.03619748793
The MAPE for Test data is 0.12940358828836765
The EVS for Test data is 0.9251368574731786
The RMSE is:269.29
```

This model is Overfit, because train results are giving very good numbers but Test RMSE value is very large as compared to Train results.

5.2 Hyper parameter tuning for random forest

5.2.1 Definition:

Hyperparameters are configurations that cannot be learnt from the regular data that we provide to the algorithm; these are inbuilt to the algorithm and each algorithm has its own predefined set of hyperparameters. Hyperparameters are often tuned for increasing model accuracy, and we can use various methods such as GridSearchCV, RandomizedSearchCV .

List of some of inbuilt hyperparameters are as follows:

- n_estimators: We know that a random forest is nothing but a group of many decision trees, the n_estimator parameter controls the number of trees inside the classifier. We may think that using many trees to fit a model will help us to get a more generalized result, The default number of estimators is 100 in scikit-learn.
- max_depth: It governs the maximum height upto which the trees inside the forest can grow. It is one of the most important hyperparameters when it comes to increasing the accuracy of the model, as we increase the depth of the tree the model accuracy increases upto a certain limit but then it will start to decrease gradually because of overfitting in the model.
- min_samples_split: It specifies the minimum amount of samples an internal node must hold in order to split into further nodes. If we have a very low value of min_samples_splits then, in this case, our tree will continue to grow and start overfitting. By increasing the value of min_samples_splits we can decrease the total number of splits thus limiting the number of parameters in the model and thus can aid in reducing the overfitting in the model. However, the value should not be kept very large that a number of parameters drop extremely causing the model to underfit. We generally keep min_samples_split value between 2 and 6. However, the default value is set to 2.
- min_samples_leaf: It specifies the minimum amount of samples that a node must hold after getting split. It also helps to reduce overfitting when we have ample amount of parameters. Less number of parameters can lead to overfitting also, we should keep in mind that increasing the value to a large number can lead to less number of parameters and in this case model can underfit also. The default value is set to 1.
- max_features: Random forest takes random subsets of features and tries to find the best split. max_features helps to find the number of features to take into account in order to make the best split. It can take four values “auto“, “sqrt“, “log2” and None.
    - In case of auto: considers max_features = sqrt(n_features)
    - In case of sqrt: considers max_features = sqrt(n_features), it is same as auto
    - In case of log2: considers max_features = log2(n_features)
    - In case of None: considers max_features = n_features
- max_leaf_nodes: It sets a limit on the splitting of the node and thus helps to reduce the depth of the tree, and effectively helps in reducing overfitting. If the value is set to None, the tree continues to grow infinitely.
- max_samples: This hyperparameter helps to choose maximum number of samples from the training dataset to train each individual tree.

These are the major hyperparameters that are present implicitly in the random forest classifier which is required to be tuned in order to increase the accuracy of our training model.

5.2.2 Action:

We have used following parameters and tested in gridSearch algorithm .

```
param_grid = {
   'max_depth': [8,10,12],
   'max_features': [3,4,5],
   'max_depth': [8,10],
   'min_samples_leaf': [30,60,90],   ## 1-3% of training data set
   'min_samples_split': [100,180,250],   ## 3 times of min sample leaf
   'n_estimators': [100,200]
}
```

Final List of parameters after testing:

```
{'max_depth': 10,
 'max_features': 5,
 'min_samples_leaf': 30,
 'min_samples_split': 100,
 'n_estimators': 200}
```

5.2.3 Feature Importance :

	Imp
1. SumAssured_Per_Policy	0.541611
2. NumberOfPolicy	0.376927
3. CustTenure	0.024193
4. Age	0.021406
5. MonthlyIncome	0.015970
6. ExistingPolicyTenure	0.010595
7. Designation	0.003730
8. ExistingProdType	0.003584
9. MaritalStatus	0.000875
10. PaymentMethod	0.000409
11. Zone	0.000235
12. EducationField	0.000172
13. Gender	0.000128
14. Occupation	0.000099
15. Channel	0.000066

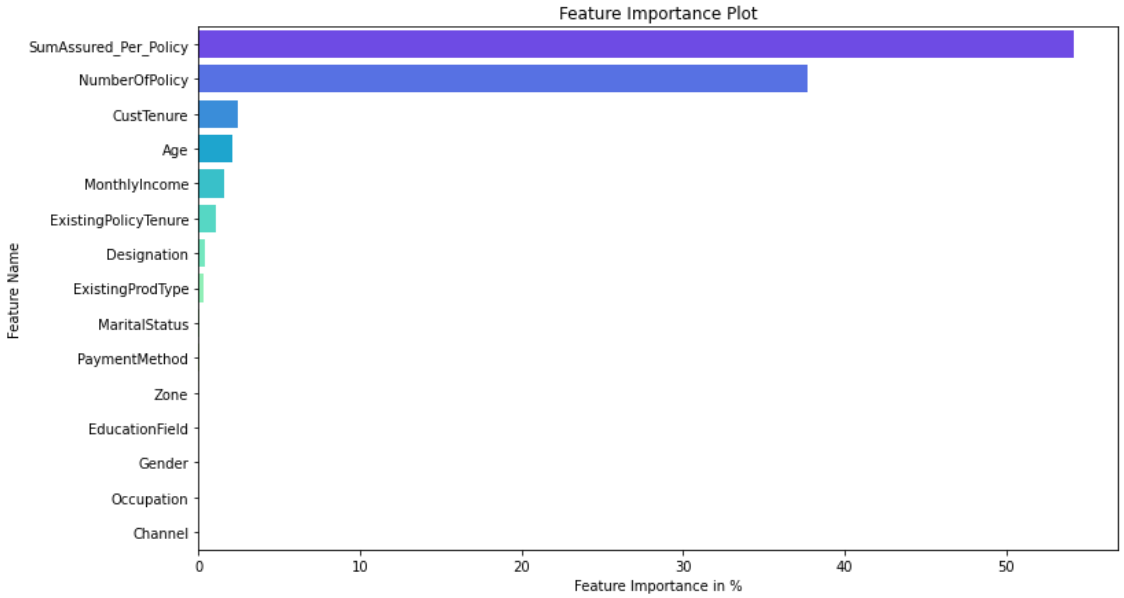


Figure 11 Best Param Random Forest Feature Importance

5.2.4 Performance parameters:

Train data matrix:

```
The MAE is: 106.47556038137378
The MSE is: 24820.70796807343
The MAPE is 0.07690297993131891
The EVS is 0.9770456925867458
The RMSE is:157.55
```

Test data matrix:

```
The MAE for Test data is: 180.50244943952802
The MSE for Test data is: 72519.03619748793
The MAPE for Test data is 0.12940358828836765
The EVS for Test data is 0.9251368574731786
The RMSE is:269.29
```

This model is Overfit, because train results are giving very good numbers but Test RMSE value is very large as compared to Train results.

5.3 Hyper parameter tuning for Decision Tree Regressor

5.3.1 Definition:

The process of reducing overfitting. Regularization is done differently in regression models. But for a decision tree classifier, we should perform regularization by providing additional arguments accepted by DecisionTreeClassifier. These arguments provided by us to revamp our model are called HyperParameters.

Hyperparameter and parameters:

Parameters are the model features that the model learns from the data. Whereas, Hyperparameters are arguments accepted by a model-making function and can be modified to reduce overfitting, leading to a better generalization of the model.

Hyperparameter List for Decision Trees:

Some of the important hyperparameters:

- Max Depth: This argument represents the maximum depth of a tree. If not specified, the tree is expanded until the last leaf nodes contain a single value. Hence by reducing this meter, we can preclude the tree from learning all training samples thereby, preventing over-fitting.
- Max leaf nodes: As the name suggests, this hyperparameter caps the number of leaf nodes in a decision tree. It will allow the branches of a tree to have varying depths, another way to control the model's complexity.
- n_estimators: This argument limits the number of decision trees in random forests. By default, its value is calibrated to 100, but in the case of larger datasets, 100 can prove to be a meager quantity. Hence, it's better to try a higher number of estimators.
- min_samples_split: Minimum samples split decides or hold the value for the minimum number of samples necessary to split a nonterminal node. By default, the decision tree tries to split every node that has two or more rows of data inside it. This can again cause memorization of training data, resulting in a lesser generalized model.
- min_samples_leaf: Minimum sample leaf may sound like minimum sample split and is somewhat similar too. But in this case, we are talking about the minimum number of samples required to be left at the leaf node. A split will only be considered if there are at least min_samples_leaf samples on the left and right branches.

5.3.2 Action:

We have used following parameters and tested in gridSearch algorithm .

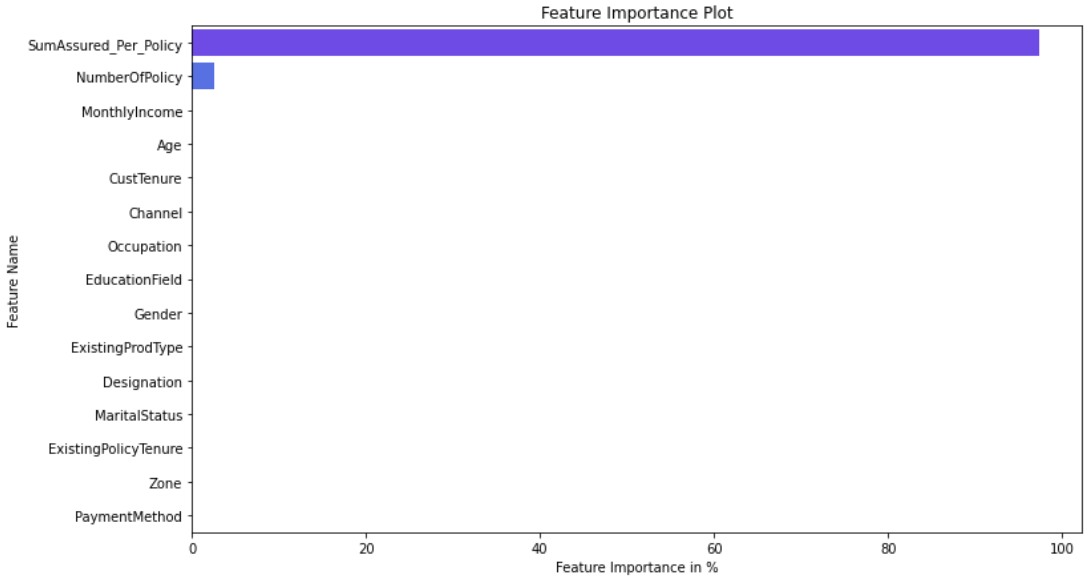
```
param_grid = {
   'criterion':  ['squared_error', 'friedman_mse', 'absolute_error'],
   'max_depth': [8,10,12],
   'min_samples_leaf': [100,150,200],
   'min_samples_split': [300,450,600],
}
```

Final List of parameters after testing:

```
DecisionTreeRegressor(max_depth=8,
                      min_samples_leaf=100,
                      min_samples_split=300,
                       random_state=1)
```

5.3.3 Feature Importance :

	Imp
SumAssured_Per_Policy	0.973447
NumberOfPolicy	0.026164
MonthlyIncome	0.000388
Age	0.000000
CustTenure	0.000000
Channel	0.000000
Occupation	0.000000
EducationField	0.000000
Gender	0.000000
ExistingProdType	0.000000
Designation	0.000000
MaritalStatus	0.000000
ExistingPolicyTenure	0.000000
Zone	0.000000
PaymentMethod	0.000000



We can clearly see that except 3 fields, there is no contribution by any other fields. Those important fields are :

SumAssured_Per_Policy	0.973447
NumberOfPolicy	0.026164
MonthlyIncome	0.000388

5.3.4 Performance parameters:

Train data matrix:

```
The MAE is: 199.28413409118212
The MSE is: 109990.86396354134
The MAPE is 0.14767656097815846
The EVS is 0.8982721849779248
The RMSE is:331.65
```

Test data matrix:

```
The MAE for Test data is: 211.17181273772886
The MSE for Test data is: 108965.47420328196
The MAPE for Test data is 0.1564641747663458
The EVS for Test data is 0.8875324431646422
The RMSE is:330.10
```

This model is Perfect fit, because Train and Test result RMSE values are and all other parameters are in about same range. This is a sign of good modelling .

6. Interpretation of the most optimum model and its implication on the business

6.1 Performance Parameters List :

We have created multiple models, and Tunned them as well by gridSearch mechanism. We have considered these perfo rmance matrixes after our model generations :

1. **MAE : Mean Absolute error :**

Absolute Error is the amount of error in your measurements. It is the difference between the measured value and “true” value. For example, if a scale states 90 pounds but you know your true weight is 89 pounds, then the scale has an absolu te error of 90 lbs – 89 lbs = 1 lbs. And Mean Absolute Error(MAE) is the average of all absolute errors. The formula is:

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n} |\boldsymbol{x}_i - \boldsymbol{x}|$$

Where:
n = the number of errors,
Σ = summation symbol (which means “add them all up”),
|xi – x| = the absolute errors.

2. **MSE (Mean squared error):**

Mean squared error (MSE) measures the amount of error in statistical models. It assesses the average squared differenc e between the observed and predicted values. When a model has no error, the MSE equals zero. As model error increas es, its value increases The formula for MSE is the following.

$$MSE = \frac{\sum(y_i - \hat{y}_i)^2}{n}$$

Where :
yi is the ith observed value.
ŷi is the corresponding predicted value.
n = the number of observations.

3. **MAPE (mean absolute percentage error):**

The mean absolute percentage error (MAPE) measures accuracy of a forecast system. It measures this accuracy as a percentage, and can be calculated as the average absolute percent error for each time period minus actual values divided by actual values.
The mean absolute percentage error (MAPE) is the most common measure used to forecast error, probably because the variable's units are scaled to percentage units, which makes it easier to understand . It works best if there are no extremes to the data (and no zeros). It is often used as a loss function in regression analysis and model evaluation.
Formula for Mean Absolute Percentage Error

$$\mathrm{M} = \frac{1}{n}\sum_{t=1}^{n}\left|\frac{A_t - F_t}{A_t}\right|$$

Where:
n is the number of fitted points,
At is the actual value,
Ft is the forecast value.
Σ is summation notation (the absolute value is summed for every forecasted point in time).

4. **EVS (Explained variance):**

Explained variance (sometimes called "explained variation") refers to the variance in the response variable in a model that can be explained by the predictor variable(s) in the model. The higher the explained variance of a model, the more t he model is able to explain the variation in the data.

5. **RMSE (Root Mean Square Error):**

Root Mean Square Error (RMSE) is the standard deviation of the residuals (prediction errors). Residuals are a measure of how far from the regression line data points are; RMSE is a measure of how spread out these residuals are. In other words, it tells you how concentrated the data is around the line of best fit. Root mean square error is commonly used in climatology, forecasting, and regression analysis to verify experimental results.
The formula is:

$$RMSE = \sqrt{\overline{(f-o)^2}}$$

Where:
f = forecasts (expected values or unknown results),
o = observed values (known results).

6.2 Gather Matrix from all Models:

We have built multiple models . List of all of these models are:

- Linear regression
- Stats Model
- Stats model Reduced parameters after VIF score and Higher P-Value
- Decision tree Model

- Random forest Model
- Random Forest Model with Bagging regressor
- Hyper parameters tuning for decision tree
- Hyper parameters tuning for Random forest

We have captured multiple performance matrix parameters, mentioned above in section 6.1 like MAE

MAE (Mean Absolute error)
MSE (Mean squared error)
MAPE (mean absolute percentage error)
EVS (Explained variance)
RMSE (Root Mean Square Error)

And we have gathered all performance parameters from all the models and put it in a data frame .

	Train RMSE	Test RMSE	Train MAE	Test MAE	Train MSE	Test MSE	Train MAPE	Test MAPE	Train EVS	Test EV
Linear Regression	317.48346	316.634998	220.321007	222.51949	100795.74739	100257.72178	0.170724	0.171265	0.906776	0.89649
Stats Model 1	317.48346002535055	316.63499771886285	220.321007	222.51949	100795.74739	100257.72178	0.170724	0.171265	0.906776	0.89649
Stats Model 2	354.54396801961036	353.56277958413546	238.541875	244.141863	125701.425259	125006.639107	0.175574	0.179678	0.883742	0.87096
Stats Model 3	354.5472293235076	353.5568690941128	238.571656	244.150994	125703.737821	125002.459684	0.175633	0.179681	0.88374	0.8709
Stats Model 4	354.57408724890837	353.35723000116565	238.559194	244.031095	125722.783348	124861.331994	0.175619	0.17965	0.883722	0.87111
Stats Model 5	354.6259691021416	353.20780003269664	238.722552	243.744927	125759.577962	124755.750004	0.175782	0.179249	0.883688	0.87122
Stats Model 6	354.6848065841537	353.0174978354055	238.752834	243.6696	125801.312022	124621.353778	0.17576	0.179242	0.883649	0.87136
Stats Model 7	354.7636336409267	352.8757950057764	238.73769	243.744927	125857.235754	124521.326701	0.175744	0.179435	0.883598	0.87146
Stats Model 8	354.8816410065518	352.68578464437667	238.760798	243.458747	125940.979124	124387.26269	0.175876	0.179265	0.88352	0.87161
Stats Model 9	355.0675639410347	352.60212402026525	238.687196	243.298407	126072.974963	124328.257864	0.175543	0.178756	0.883398	0.87167
Decision Tree	0.0	369.334449	0.0	239.356932	0.0	136407.935547	0.0	0.173311	1.0	0.85928
Random Forest	101.955542	270.907949	65.695396	178.33923	10394.932541	73391.11693	0.047686	0.128212	0.990386	0.92423
ANN Model	5082.681797	5093.46129	4972.297786	4990.048298	25833654.247346	25943347.91466	5.255689	5.291988	-0.092953	-0.09990
Random Forest Bagging Model	157.545892	269.293587	106.47556	180.502449	24820.707968	72519.036197	0.076903	0.129404	0.977046	0.92513
Random Forest Tunned Param	157.545892	269.293587	106.47556	180.502449	24820.707968	72519.036197	0.076903	0.129404	0.977046	0.92513
decission_tree tunned param	331.648706	330.099188	199.284134	211.171813	109990.863964	108965.474203	0.147677	0.156464	0.898272	0.88753

Comparison:

There are multiple ways we can compare our models. I have done comparison with Lowest RMSE values and Lower MAPE values.

RMSE gives us Error in the model and MAPE gives good result in percentage wise.

Lowest RMSE:

	Train RMSE	Test RMSE	Train MAE	Test MAE	Train MSE	Test MSE	Train MAPE	Test MAPE	Train EVS	Test EVS
Random Forest Bagging Model	157.545892	269.293587	106.47556	180.502449	24820.707968	72519.036197	0.076903	0.129404	0.977046	0.925137
Random Forest Tunned Param	157.545892	269.293587	106.47556	180.502449	24820.707968	72519.036197	0.076903	0.129404	0.977046	0.925137
Random Forest	101.955542	270.907949	65.695396	178.33923	10394.932541	73391.11693	0.047686	0.128212	0.990386	0.924232
Linear Regression	317.483460	316.634998	220.321007	222.51949	100795.74739	100257.72178	0.170724	0.171265	0.906776	0.896495
Stats Model 1	317.483460	316.634998	220.321007	222.51949	100795.74739	100257.72178	0.170724	0.171265	0.906776	0.896495
decission_tree tunned param	331.648706	330.099188	199.284134	211.171813	109990.863964	108965.474203	0.147677	0.156464	0.898272	0.887532
Stats Model 9	355.067564	352.602124	238.687196	243.298407	126072.974963	124328.257864	0.175543	0.178756	0.883398	0.871676
Stats Model 8	354.881641	352.685785	238.760798	243.458747	125940.979124	124387.26269	0.175876	0.179265	0.88352	0.871611
Stats Model 7	354.763634	352.875795	238.73769	243.744927	125857.235754	124521.326701	0.175744	0.179435	0.883598	0.871469
Stats Model 6	354.684807	353.017498	238.752834	243.6696	125801.312022	124621.353778	0.17576	0.179242	0.883649	0.871367
Stats Model 5	354.625969	353.207800	238.722552	243.744927	125759.577962	124755.750004	0.175782	0.179249	0.883688	0.871228
Stats Model 4	354.574087	353.357230	238.559194	244.031095	125722.783348	124861.331994	0.175619	0.17965	0.883722	0.871115
Stats Model 3	354.547229	353.556869	238.571656	244.150994	125703.737821	125002.459684	0.175633	0.179681	0.88374	0.87097
Stats Model 2	354.543968	353.562780	238.541875	244.141863	125701.425259	125006.639107	0.175574	0.179678	0.883742	0.870965
Decision Tree	0.000000	369.334449	0.0	239.356932	0.0	136407.935547	0.0	0.173311	1.0	0.859284
ANN Model	5082.681797	5093.461290	4972.297786	4990.048298	25833654.247346	25943347.91466	5.255689	5.291988	-0.092953	-0.099901

Figure 12 Sort performance matrixes with Lowest RMSE

Insights: With Lowest RMSE Random forest Model with Bagging regressor and Random forest model with Tunned Hyper Parameters, both giving same results. But there are big difference in train and test values of RMSE, which indicates overfitting of the model. In this case, we will consider 4th Lowest RMSE Model is the best one, which is "Linear Regression" Model. It has about same values of RMSE value

Lowest MAPE:

	Train RMSE	Test RMSE	Train MAE	Test MAE	Train MSE	Test MSE	Train MAPE	Test MAPE	Train EVS	Test EVS
Random Forest	101.955542	270.907949	65.695396	178.33923	10394.932541	73391.11693	0.047686	0.128212	0.990386	0.924232
Random Forest Bagging Model	157.545892	269.293587	106.47556	180.502449	24820.707968	72519.036197	0.076903	0.129404	0.977046	0.925137
Random Forest Tunned Param	157.545892	269.293587	106.47556	180.502449	24820.707968	72519.036197	0.076903	0.129404	0.977046	0.925137
decission_tree tunned param	331.648706	330.099188	199.284134	211.171813	109990.863964	108965.474203	0.147677	0.156464	0.898272	0.887532
Stats Model 1	317.483460	316.634998	220.321007	222.51949	100795.74739	100257.72178	0.170724	0.171265	0.906776	0.896495
Linear Regression	317.483460	316.634998	220.321007	222.51949	100795.74739	100257.72178	0.170724	0.171265	0.906776	0.896495
Decision Tree	0.000000	369.334449	0.0	239.356932	0.0	136407.935547	0.0	0.173311	1.0	0.859284
Stats Model 9	355.067564	352.602124	238.687196	243.298407	126072.974963	124328.257864	0.175543	0.178756	0.883398	0.871676
Stats Model 6	354.684807	353.017498	238.752834	243.6696	125801.312022	124621.353778	0.17576	0.179242	0.883649	0.871367
Stats Model 5	354.625969	353.207800	238.722552	243.744927	125759.577962	124755.750004	0.175782	0.179249	0.883688	0.871228
Stats Model 8	354.881641	352.685785	238.760798	243.458747	125940.979124	124387.26269	0.175876	0.179265	0.88352	0.871611
Stats Model 7	354.763634	352.875795	238.73769	243.744927	125857.235754	124521.326701	0.175744	0.179435	0.883598	0.871469
Stats Model 4	354.574087	353.357230	238.559194	244.031095	125722.783348	124861.331994	0.175619	0.17965	0.883722	0.871115
Stats Model 2	354.543968	353.562780	238.541875	244.141863	125701.425259	125006.639107	0.175574	0.179678	0.883742	0.870965
Stats Model 3	354.547229	353.556869	238.571656	244.150994	125703.737821	125002.459684	0.175633	0.179681	0.88374	0.87097
ANN Model	5082.681797	5093.461290	4972.297786	4990.048298	25833654.247346	25943347.91466	5.255689	5.291988	-0.092953	-0.099901

Figure 13Sort Performance matrixes with lowest MAPE value

Insights:

Random forest related all 3 models have lowest MAPE model. But again, those seems over fitting the model, because train and test values have big difference.

Whereas Decision tree with tunned hyper parameters , Linear Regression and Stats Model seems best model among all , which are giving similar results for both train and test data set. Giving 14.7% for decision tree and 17 % for Linear and stats model of MAPE value.