

Business Report

Capstone Project

LI_BFSI_01+Life+Insurance+Sales

Final Report submission

Date: 20-Jan-2023

Created by Amit Jain

Table of Contents

List of Figure	5
1. Introduction of the business Problem.....	6
1.1 Defining Problem Statement:	6
1.2 Need of the study/project:	6
2. Data Export :	6
2.1 Understanding how data was collected in terms of time, frequency and methodology:	6
2.2 Visual inspection of data (rows, columns, descriptive details):	7
2.3 Understanding of attributes (variable info, renaming if required):	8
3. Exploratory data analysis	9
3.1 Addition of new variables (if required).....	9
3.2 Missing Value treatment (if applicable)	10
3.3 Duplicate checks:.....	11
3.4 Removal of unwanted variables (if applicable).....	12
3.5 Outlier removal (if applicable):.....	13
3.6 Univariate Analysis.....	14
3.6.1 Taking counts for each Categorical field :.....	14
3.6.2 Merge similar categorical values:	15
3.6.3 Taking counts for each Categorical fields after merging similar values:	16
3.6.4 Generate Histogram and Boxplot for Sample data:.....	18
3.6.5 Check data skewness :	19
3.7 Bivariate Analysis	20
3.8 Multivariate Analysis.....	25
4. Variable transformation / Data Transformations and Scaling	27
4.1 Numeric Variable Transformation:	28
4.2 Categorical Variable Transformation (Encoding)	28
5. Data Split: Split the data into train and test (70:30)	30
6. Model building and interpretation :.....	31
6.1 SKLearn Linear Model:	31
6.1.1 Definition:	31
6.1.2 Action:.....	32
6.1.3 Equation:	32
6.1.4 Understanding Model:	33

6.1.5	Performance parameters:	34
6.1.6	Feature importance	34
6.2	Building Linear Model with Scaled parameters :.....	34
6.2.1	Definition:	34
6.2.2	Action:.....	35
6.2.3	Equation:	35
6.2.4	Performance parameters:	35
6.3	Building Stats Model:	36
6.3.1	Definition:	36
6.3.2	Action:.....	36
6.3.3	Performance parameters:	38
6.3.4	Feature Importance:	38
6.4	Building Multiple Stats Model to eliminate non affecting fields :.....	39
6.4.1	Stats Model for fields with lower value of VIF :	39
6.4.1.1	How to interpret a given VIF value?:.....	39
6.4.1.2	Action:.....	40
6.4.1.3	Performance parameters:	41
6.4.1.4	Stats Model for fields with Lower P value and remove other fields:	42
6.4.1.5	Action:.....	42
6.4.1.6	Performance parameters:	43
6.4.1.7	Equation and interpretation:	44
6.5	Decision Tree Model	44
6.5.1	Definition:	44
6.5.2	Action:.....	45
6.5.3	Feature Importance :.....	45
6.5.4	Performance parameters:	46
6.6	Random Forest Regressor Model.....	46
6.6.1	Definition :	46
6.6.2	Action:.....	47
6.6.3	Feature Importance :.....	47
6.6.4	Performance parameters:	48
6.7	Artificial Neural Network Regressor	48
6.7.1	Definition:	48

6.7.2	Action:.....	49
6.7.3	Feature Importance :.....	49
6.7.4	Performance parameters:	50
7.	Model Tuning and business implication	50
7.1	Bagging Regressor (Random Forest should be applied for Bagging)	50
7.1.1	Definition :	50
7.1.2	Action:.....	51
7.1.3	Feature Importance :.....	51
7.1.4	Performance parameters:	51
7.2	Hyper parameter tuning for random forest	52
7.2.1	Definition:	52
7.2.2	Action:.....	53
7.2.3	Feature Importance :.....	53
7.2.4	Performance parameters:	54
7.3	Hyper parameter tuning for Decision Tree Regressor	54
7.3.1	Definition:	54
7.3.2	Action:.....	55
7.3.3	Feature Importance :.....	55
7.3.4	Performance parameters:	56
8.	Interpretation of the most optimum model and its implication on the business	56
8.1	Performance Parameters List :.....	56
8.2	Gather Matrix from all Models:	57
9.	Business insights , Recommendations	60
9.1	Is the data unbalanced? If so, what can be done?:	60
9.2	Business insights using clustering	61
9.3	Business insights from EDA	62
9.4	Business benefits:	63
9.5	Improvement Area and recommendations:	63

List of Figure

Figure 1 Boxplot	13
Figure 2 : Histogram and Boxplot.....	18
Figure 3 Mean, Count and Sum graph for categorical fields	20
Figure 4 heat Map.....	26
Figure 5 pair plot against AgentBonus per policy	27
Figure 6 Scatter plot for Prediction on Test data Figure 7 Scatter plot for Prediction on Train data	33
Figure 8 Scatter plot for Prediction on Scaled Test data Figure 9 Scatter plot for Prediction on Scaled train data.....	35
Figure 10 Scatter Plot for Predicted values on Test data set.....	38
Figure 11 Example for Decision tree	44
Figure 12 Decision Tree Feature Importance.....	45
Figure 13 Example for Random Forest.....	46
Figure 14 Random Forest Feature Importance.....	47
Figure 15 Example for ANN Model	49
Figure 16 Explain Bagging	51
Figure 17 Explain Bootstrapping	51
Figure 18 Best Param Random Forest Feature Importance	53
Figure 19 Best feature Decision Tree Tunned Param	55
Figure 20 Sort performance matrixes with Lowest RMSE	59
Figure 21 Sort Performance matrixes with lowest MAPE value	60

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.

2. Data Export :

2.1 Understanding how data was collected in terms of time, frequency and methodology:

Assume that the data follows a normal distribution. In reality, the normality assumption may not always hold if the sample size is small.

Also we have not given specific time frame, when this data elements were collected and how frequently this kind of information is being collected. Though we assume this all sample data belong to same time frame window for consigning it as a Randomly collected records, without any influence .

2.2 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:

	CustID	AgentBonus	Age	CustTenure	Channel	Occupation	EducationField	Gender	ExistingProdType	Designation	NumberOfPolicy	MaritalStatus	MonthlyIncome
0	7000000	4409	22.0	4.0	Agent	Salaried	Graduate	Female	3	Manager	2.0	Single	10000
1	7000001	2214	11.0	2.0	Third Party Partner	Salaried	Graduate	Male	4	Manager	4.0	Divorced	15000
2	7000002	4273	26.0	4.0	Agent	Free Lancer	Post Graduate	Male	4	Exe	3.0	Unmarried	20000
3	7000003	1791	11.0	NaN	Third Party Partner	Salaried	Graduate	Female	3	Executive	3.0	Divorced	12000
4	7000004	2955	6.0	NaN	Agent	Small Business	UG	Male	3	Executive	4.0	Divorced	8000

Data dictionary:

CustID =>Unique customer ID

AgentBonus =>Bonus amount given to each agents in last month

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

Complaint =>Indicator of complaint registered in last one month by customer

ExistingPolicyTenure =>Max tenure in all existing policies of customer

SumAssured =>Max of sum assured in all existing policies of customer

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

LastMonthCalls =>Total calls attempted by company to a customer for cross sell

CustCareScore =>Customer satisfaction score given by customer in previous service call

2.3 Understanding of attributes (variable info, renaming if required):

Data description:

	count	unique	top	freq	mean	std	min	25%	50%	75%	max
CustID	4520.0	NaN	NaN	NaN	7002259.5	1304.955938	7000000.0	7001129.75	7002259.5	7003389.25	7004519.0
AgentBonus	4520.0	NaN	NaN	NaN	4077.838274	1403.321711	1605.0	3027.75	3911.5	4867.25	9608.0
Age	4251.0	NaN	NaN	NaN	14.494707	9.037629	2.0	7.0	13.0	20.0	58.0
CustTenure	4294.0	NaN	NaN	NaN	14.469027	8.963671	2.0	7.0	13.0	20.0	57.0
Channel	4520	3	Agent	3194	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Occupation	4520	5	Salaried	2192	NaN	NaN	NaN	NaN	NaN	NaN	NaN
EducationField	4520	7	Graduate	1870	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Gender	4520	3	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	6	Manager	1620	NaN	NaN	NaN	NaN	NaN	NaN	NaN
NumberOfPolicy	4475.0	NaN	NaN	NaN	3.565363	1.455926	1.0	2.0	4.0	5.0	6.0
MaritalStatus	4520	4	Married	2268	NaN	NaN	NaN	NaN	NaN	NaN	NaN
MonthlyIncome	4284.0	NaN	NaN	NaN	22890.309991	4885.600757	16009.0	19683.5	21606.0	24725.0	38456.0
Complaint	4520.0	NaN	NaN	NaN	0.287168	0.452491	0.0	0.0	0.0	1.0	1.0
ExistingPolicyTenure	4336.0	NaN	NaN	NaN	4.130074	3.346386	1.0	2.0	3.0	6.0	25.0
SumAssured	4366.0	NaN	NaN	NaN	619999.699267	246234.82214	168536.0	439443.25	578976.5	758236.0	1838496.0
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
LastMonthCalls	4520.0	NaN	NaN	NaN	4.626991	3.620132	0.0	2.0	3.0	8.0	18.0
CustCareScore	4468.0	NaN	NaN	NaN	3.067592	1.382968	1.0	2.0	3.0	4.0	5.0

Insights:

1. Agent Bonus : Bonus given to Agents, as well as Target variable. Minimum bonus given as 1605 and Max is 9608
2. Age: Customers of all age group from 2 years to 58 years, there are also customers with 0 Age, needs to be corrected
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 , if it is blank, that needs to be corrected
6. Monthly income Ranging between about 16K to 38K , if it is blank, needs to be corrected
7. Complaint: 0 to 1 complaint in last one month
8. Gender showing as 3 ,: this needs to be corrected
9. Occupation : Different Occupations, there can be multiple business with same name, needs to be corrected
10. Education : can have similar names of degrees, needs to be corrected
11. Designation: can be duplicate
12. existing Policy tenure. if it is blank needs to be corrected
13. Sum Assured : if it is blank needs to be corrected
14. Customer care score. needs to be corrected if it is blank

Data info:

#	Column	Non-Null	Count	Dtype
0	CustID	4520	non-null	int64
1	AgentBonus	4520	non-null	int64
2	Age	4251	non-null	float64
3	CustTenure	4294	non-null	float64
4	Channel	4520	non-null	object
5	Occupation	4520	non-null	object
6	EducationField	4520	non-null	object
7	Gender	4520	non-null	object
8	ExistingProdType	4520	non-null	int64
9	Designation	4520	non-null	object
10	NumberOfPolicy	4475	non-null	float64
11	MaritalStatus	4520	non-null	object
12	MonthlyIncome	4284	non-null	float64
13	Complaint	4520	non-null	int64
14	ExistingPolicyTenure	4336	non-null	float64
15	SumAssured	4366	non-null	float64
16	Zone	4520	non-null	object
17	PaymentMethod	4520	non-null	object
18	LastMonthCalls	4520	non-null	int64
19	CustCareScore	4468	non-null	float64

Insights:

1. Customer id can be removed from data, as it will not be required for Bonus prediction
2. Channel , occupation, Education, Gender, Designation , Marital Status and all other Object type fields should be converted to Numeric format in order to use it in prediction Model.

3. Exploratory data analysis

3.1 Addition of new variables (if required)

AgentBonus is the amount of Bonus given to Insurance agents based, what policy he/she has sold to customers, and what profit Company might have taken. Since one customer can buy multiple policies together, so it will not be fare to Compare AgentBonus amount of one agent with another.

May be AgentBonus for one agent is more (because hist Customer bought 5 policies together). but Average of per policy of that bonus amount can be lower than Single Policy of bigger SumAssured.

- So lets create additional Field, "AgentBonus_Per_Policy", which should be calculated by:

$$\text{AgentBonus_Per_Policy} = \text{AgentBonus} / \text{NumberOfPolicy}$$

- Also lets create a new field called SumAssured_Per_Policy, which should be calculated by :

$$\text{SumAssured_Per_Policy} = \text{SumAssured} / \text{NumberOfPolicy}$$

Now we have got this list of fields, which in new data set:

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

whereas Old data set fields are:

```
Index(['CustID', 'AgentBonus', 'Age', 'CustTenure', 'Channel', 'Occupation',  
      'EducationField', 'Gender', 'ExistingProdType', 'Designation',  
      'NumberOfPolicy', 'MaritalStatus', 'MonthlyIncome', 'Complaint',  
      'ExistingPolicyTenure', 'SumAssured', 'Zone', 'PaymentMethod',  
      'LastMonthCalls', 'CustCareScore'],  
      dtype='object')
```

We have got 2 new fields , in our data set.

3.2 Missing Value treatment (if applicable)

We have taken NULL counts for all of our attributes and NULL value counts are as follows:

CustID	0
AgentBonus	0
Age	269
CustTenure	226
Channel	0
Occupation	0
EducationField	0
Gender	0
ExistingProdType	0
Designation	0
NumberOfPolicy	45
MaritalStatus	0
MonthlyIncome	236
Complaint	0
ExistingPolicyTenure	184
SumAssured	154
Zone	0
PaymentMethod	0
LastMonthCalls	0
CustCareScore	52
AgentBonus_Per_Policy	45
SumAssured_Per_Policy	199
dtype:	int64

Total NULL values in data : 1410

Total data elements in Sample data are: 99440

1410 is the total Null counts, which includes NULL counts for our 2 newly created fields as well, which are :

AgentBonus_Per_Policy	45
SumAssured_Per_Policy	199

If we subtract this count, then actual total number of missing values count is 1166

NULL Treatment one by one for each NULL field:

➤ Treating NULL Age:

Age should be Greater than or Equals to Customer Tenure as well as Existing Policy Tenure. So let's give first preference to Customer Tenure and then existing Policy Tenure for correcting NULL "Age"

There were total 269 NULL records in Age field. We will consider Policy Started , as child gets Birth and Age should be replaced with "CustomerTenure " and then "ExistingPolicyTenure".

➤ Treating NULL CustomerTenure:

We will replace NULL "CustomerTenure " field with "ExistingPolicyTenure" and if "ExistingPolicyTenure" is also not available then we should replace "CustomerTenure " with "Age" field.

➤ Treating "Existing Policy tenure":

Similarly, if Cust "Existing Policy tenure" is NULL, then replace it with "Customer Tenure" and second preference will be to replace it with "Age"

➤ Treating "NumberOfPolicy":

Best way to Treat this field is to replace NULL with Mode.

➤ Treating NULLs for "AgentBonus_Per_Policy":

Since we have already computed NumberOfPolicy Field, then for treating missing values for "AgentBonus_Per_Policy", we will again use our formula for this field.

```
df2['AgentBonus_Per_Policy']=df2['AgentBonus']/df2['NumberOfPolicy']
```

and NULL has been treated.

➤ Treating NULL for "MonthlyIncome" and "SumAssured":

Best way to Treat this field is to use KNN Imputers for missing NULL values.

I have used KNNImputer from SKLearn library and used this method for imputing missing values for "MonthlyIncome" and "SumAssured".

➤ Treating NULL for "CustCareScore":

I have used Median for this value and treating missing values for "CustCareScore".

Now we are left with only SumAssured field , which have NULL values, but we will impute this, as we are using SumAssured_per_policy field instead, which we have already corrected and we are going to drop field SumAssured in our next Step.

3.3 Duplicate checks:

We checked duplicity in CustID field, and we did not find any Duplicate records. Total Count for Duplicate records are : 0

3.4 Removal of unwanted variables (if applicable)

Since We have used AgentBonus and SumAssured for building our new field, so we don't need them any more for our analysis, Also We dont need field CustId for our analysis, as it will not add any benefits in our analysis. So we will drop these 3 fields from our data:

AgentBonus , SumAssured , CustID

And we have added these 2 fields in our data :

AgentBonus_per_policy , SumAssured_per_policy

We will also check for co-relations, in further sections and if we find that any field don't have relation with Target field , then we will drop those fields as well.

	Complaint	LastMonthCalls	CustCareScore
Age	0.019496	0.123837	0.029853
CustTenure	0.00685	0.142982	0.013919
ExistingProdType	-0.003486	0.033191	0.00411
NumberOfPolicy	-0.016014	0.075138	-0.001005
MonthlyIncome	-0.004815	0.34393	0.035751
Complaint	1	-0.02632	-0.003814
ExistingPolicyTenure	-0.005082	0.126951	0.013532
LastMonthCalls	-0.02632	1	0.006386
CustCareScore	-0.003814	0.006386	1
AgentBonus_Per_Policy	0.025091	0.038717	-0.005319
SumAssured_Per_Policy	0.023838	0.03474	-0.013488

We have built co-relation matrix for all the fields and checked it's relation with Targeted field. This is co-relation with AgentBonus:

Complaint : 0.025091

LastMonthCalls: 0.038717

CustCareScore: -0.005319

we have checked that following fields have very minute impact on targeted fields and these can be dropped as well:

3.5 Outlier removal (if applicable):

We have analyzed data from the boxplot:

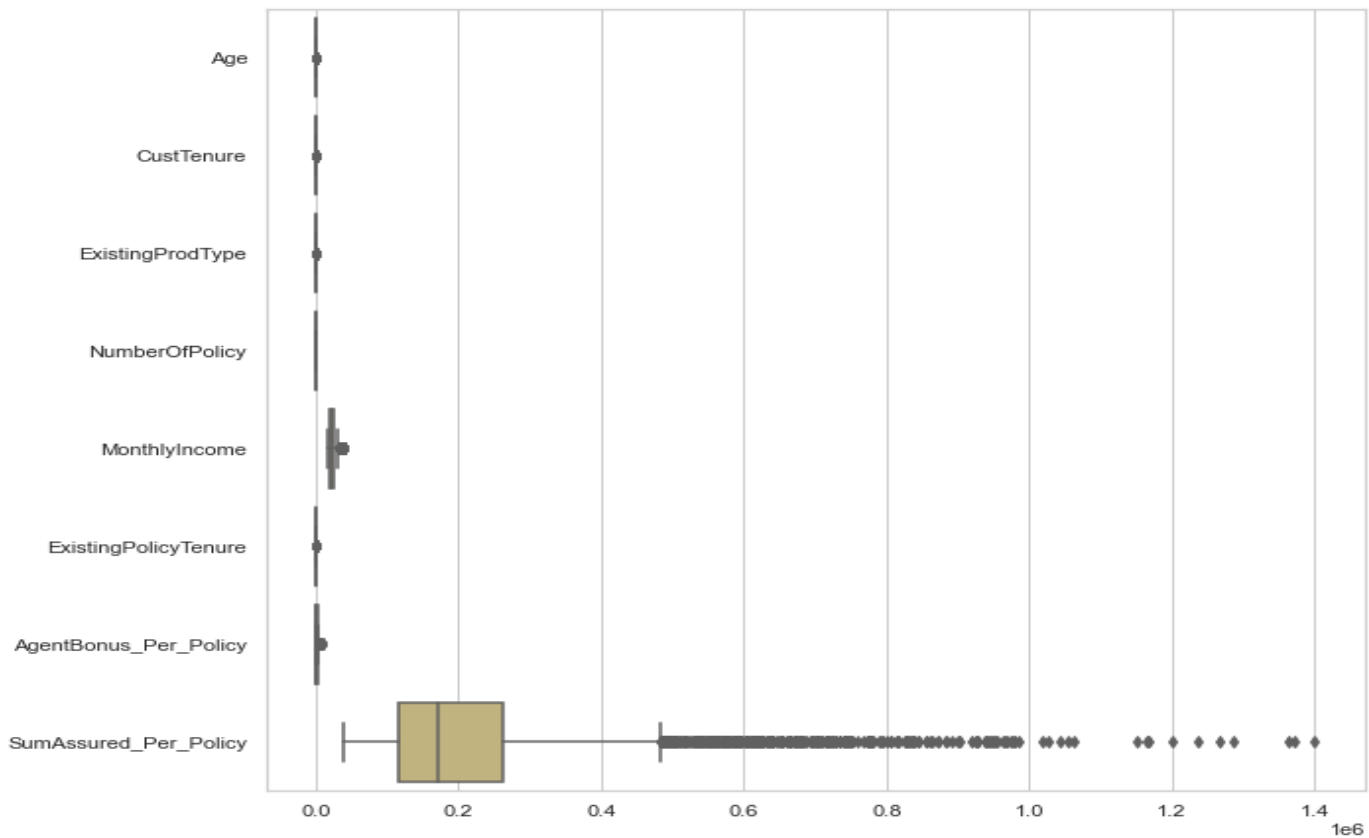


Figure 1 Boxplot

Action :

We do see outliers in about each of the field of sample data, but data scales are different,

Example: Age have limited discrete values, AgentBonus will be always lesser than SumAssured and it will be just a fraction of percentage of SumAssured . Customer Tenure, Existing Policy tenure, and other Age type fields are also discrete in nature.

So I would not like to treat outliers, as People have different income slaves and can take multiple or Single Policy as well. **And I really want to understand nature of other fields, how they are affecting our target field, AgentBonus , so we will not treat the outliers**

3.6 Univariate Analysis

3.6.1 Taking counts for each Categorical field :

We have taken counts for all the Categorical fields in Sample data set and this is the result for it :

```
Field name is CHANNEL : and Count for this categories are 3
Online          468
Third Party Partner  858
Agent           3194
Name: Channel, dtype: int64
```

```
Field name is OCCUPATION : and Count for this categories are 5
Free Lancer      2
Laarge Business  153
Large Business   255
Small Business   1918
Salaried         2192
Name: Occupation, dtype: int64
```

```
Field name is EDUCATIONFIELD : and Count for this categories are 7
MBA              74
UG               230
Post Graduate    252
Engineer         408
Diploma          496
Under Graduate   1190
Graduate         1870
Name: EducationField, dtype: int64
```

```
Field name is GENDER : and Count for this categories are 3
Fe male         325
Female          1507
Male            2688
Name: Gender, dtype: int64
```

```
Field name is DESIGNATION : and Count for this categories are 6
Exe             127
VP              226
AVP             336
Senior Manager  676
Executive       1535
Manager         1620
Name: Designation, dtype: int64
```

```
Field name is MARITALSTATUS : and Count for this categories are 4
Unmarried       194
Divorced        804
Single          1254
Married         2268
Name: MaritalStatus, dtype: int64
```

```
Field name is  ZONE : and Count for this categories are  4
South          6
East           64
North         1884
West          2566
Name: Zone, dtype: int64
```

```
Field name is  PAYMENTMETHOD : and Count for this categories are  4
Quarterly      76
Monthly        354
Yearly         1434
Half Yearly    2656
Name: PaymentMethod, dtype: int64
```

Insights:

1. Channel as "Agent" has maximum count of 3194, and least no of channel is Online. with 468
2. OCCUPATION : Maximum Policy Holder are Salaried Employees, where as free Lancer and Large business people are very less in Insurance .
3. OCCUPATION : "Laarge Business" and "Large Business" are same, it needs to be merged.
4. EDUCATIONFIELD: "UG" and "Under Graduate" are same category and needs to merged.
5. EDUCATIONFIELD: Highest Policy holders are Graduate and Least are MBA degree Holder.
6. GENDER : "Fe male" and "Female" are same, and needs to merge into one category
7. Maximum policy Holders are male with 2688 Count.
8. DESIGNATION : Exe and Executive are same and can be merge into one Category.
9. South Zone and followed by east zone have very less number of customers , need to work in that area
10. Customers like to pay half yearly and yearly payments method most for the payment of their premiums.

3.6.2 Merge similar categorical values:

From taking counts of each Categorical field, we have seen that , there are some values, which we can merge into Single value.

Example:

OCCUPATION : "Laarge Business" and "Large Business can be merged into Single field “Large business”
 EDUCATIONFIELD: "UG" and "Under Graduate" can be merged into “Under graduate”
 GENDER : "Fe male" and "Female" are same and should be merged into “Female”
 DESIGNATION : Exe and Executive are same and should be merged into “Executive”

First of all take the counts for above listed 4 Categorical field and check before merger counts:

```
Name: EducationField, dtype: int64
Graduate          1870
Under Graduate    1190
Diploma           496
Engineer          408
Post Graduate     252
UG                230
MBA               74
```

```
Name: Gender, dtype: int64
Male      2688
Female    1507
Fe male   325
```

```
Name: Occupation, dtype: int64
Salaried      2192
Small Business 1918
Large Business 255
Laarge Business 153
Free Lancer    2
```

```
Name: Designation, dtype: int64
Manager      1620
Executive    1535
Senior Manager 676
AVP          336
VP           226
Exe          127
```

Above highlighted in yellow field should be merged . we have performed merging of above listed fields

3.6.3 Taking counts for each Categorical fields after merging similar values:

After performing merging of similar values, this is how the counts look like :

```
Field name is CHANNEL : and Count for this categories are 3
Online      468
Third Party Partner 858
Agent       3194
Name: Channel, dtype: int64
```

```
Field name is OCCUPATION : and Count for this categories are 4
Free Lancer    2
Large Business 408
Small Business 1918
Salaried       2192
```

```
Field name is EDUCATIONFIELD : and Count for this categories are 6
MBA           74
Post Graduate 252
Engineer      408
Diploma       496
Under Graduate 1420
Graduate      1870
```

```
Field name is GENDER : and Count for this categories are 2
Female      1832
Male        2688
```

```
Field name is DESIGNATION : and Count for this categories are 5
VP          226
AVP         336
Senior Manager 676
Manager     1620
Executive   1662
```


Field name is MARITALSTATUS : and Count for this categories are 4

Unmarried	194
Divorced	804
Single	1254
Married	2268

Field name is ZONE : and Count for this categories are 4

South	6
East	64
North	1884
West	2566

Field name is PAYMENTMETHOD : and Count for this categories are 4

Quarterly	76
Monthly	354
Yearly	1434
Half Yearly	2656

Insights:

1. Maximum number of policies sold by Agents and very less policies sold by Online channel.
2. Maximum number of policy holders are Salaried with count of 2192 records and least number of Policy holders are free Lancer.
3. Maximum number of policy holders are Graduate in Education with count of 1870 records, and MBA holders are least with no of 74.
4. Male are Maximum number of policy holders as compared to female records.
5. Married people are Maximum number of policy holders whereas unmarried people don't prefer taking Insurance policy.
6. There are maximum policies sold in West region of the country, whereas South region have very less 6 policies sold.
7. Maximum number of policy holders prefer to pay Half yearly premiums .

3.6.4 Generate Histogram and Boxplot for Sample data:

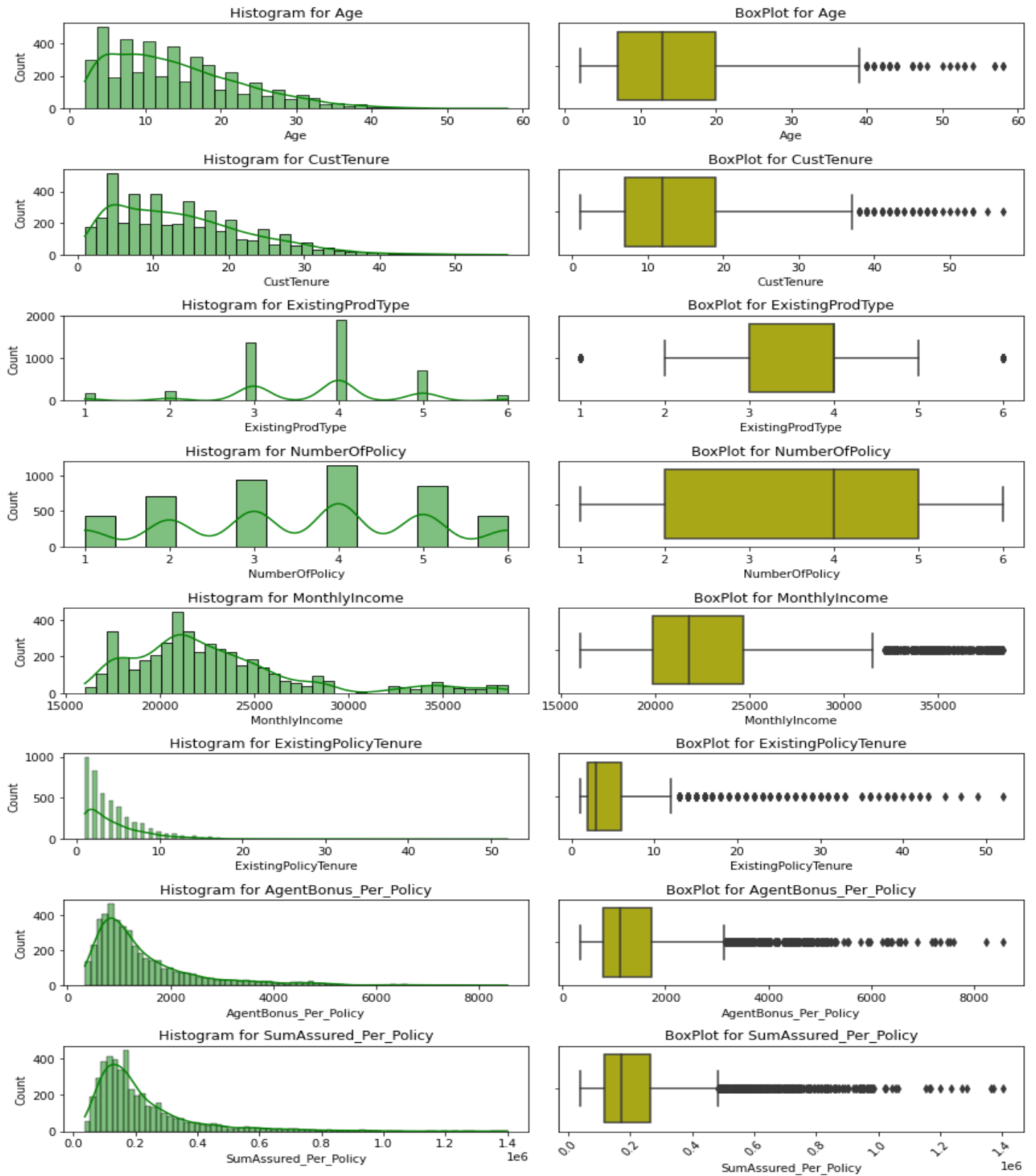


Figure 2 : Histogram and Boxplot

Insights:

1. Data is not 100% normally distributed .
 - a. Age: we can have any range of Age for customer, maximum Age is 58 years in data, which is valid.
 - b. CustomerTenure is about similar to Age field, CustomerTenure can also range from 1 year to Maximum of Age limit of the customer.
 - c. Number of Policy is a discrete field, which have 6 possible values in it, maximum customers have taken 4 policies.
 - d. Monthly Income field is a continuous field, and its ranging from 15K to about 40k , which is normal. Data is right Skewed.
 - e. Existing Policy tenure shows maximum Customer are part of under 10 years and there are a very few customers from 10 to 58 years of existing Policies.
 - f. AgentBonus_per_policy and SumAssured_per_policy is Continuous fields, and have similar distribution of data , this data is also right skewed.
2. About all the fields have some outliers.

3.6.5 Check data skewness :

```
Age                0.960101
CustTenure         0.928995
ExistingProdType   -0.401100
NumberOfPolicy     -0.108161
MonthlyIncome      1.373508
ExistingPolicyTenure 3.440053
AgentBonus_Per_Policy 2.144651
SumAssured_Per_Policy 2.373659
dtype: float64
```

Insights:

1. data is not 100% Normally distributed
2. About all data have Outliers, and which is possible because , insurance Policies , can be different for many customers, depending on their Need , worth and sum insured.
3. All data is Slightly Right Skewed .

3.7 Bivariate Analysis

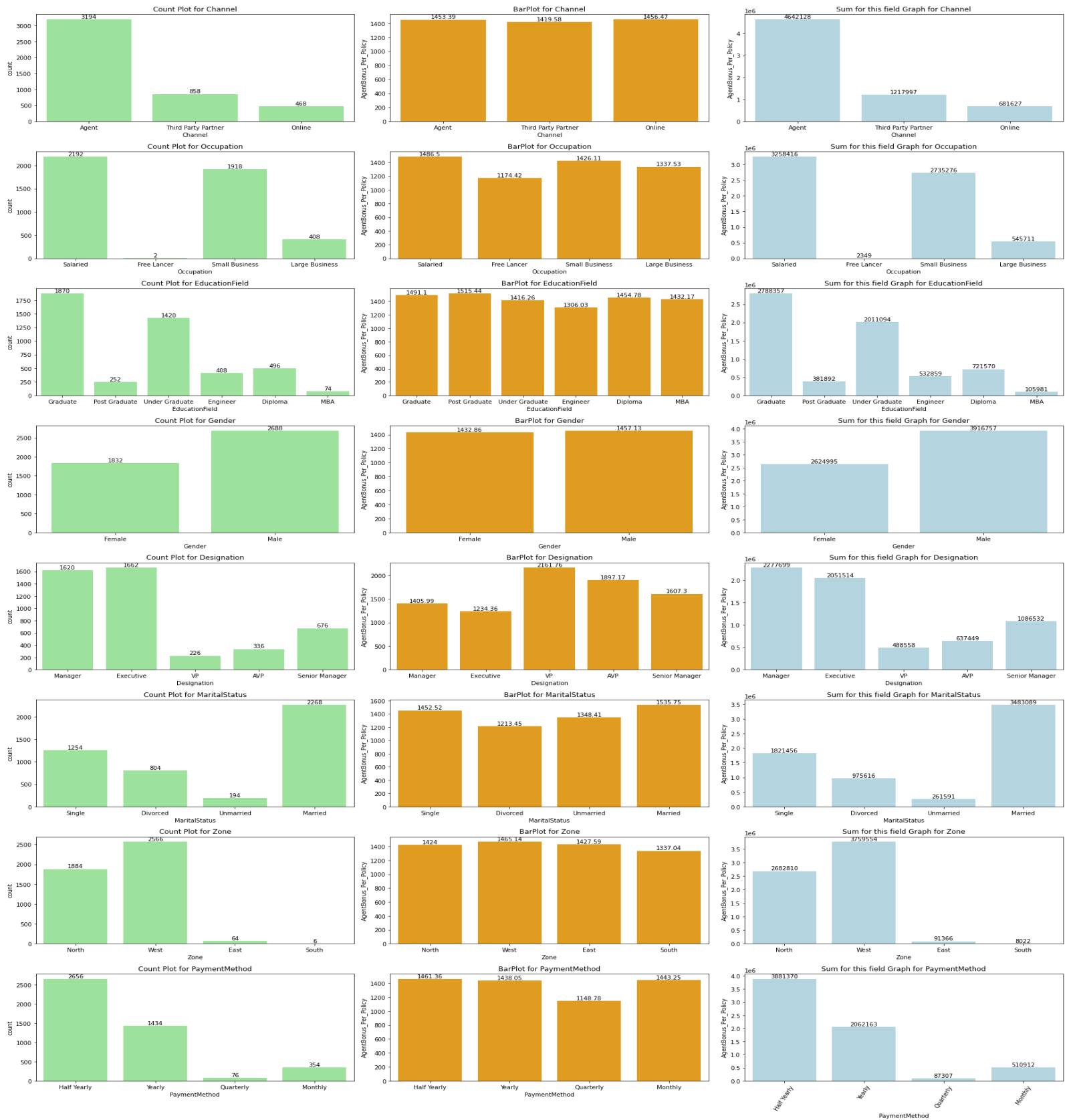


Figure 3 Mean, Count and Sum graph for categorical fields

I have performed, Bivariate analysis of all other the fields, with respect to Agent Bonus per policy as this is our target variable, and take Mean, Counts and Sum of Categorical fields.

Field name is CHANNEL : and Distinct Count for this Column is 3
Field name is CHANNEL : and it's distinct categories are ['Agent' 'Third Party Partner' 'Online'] And Counts are each categories of this column are :

Online	468
Third Party Partner	858
Agent	3194

Name: Channel, dtype: int64

Average AgentBonus for each categories of this Column are: Channel

Third Party Partner	1419.576690
Agent	1453.390106
Online	1456.468768

Name: AgentBonus_Per_Policy, dtype: float64

Total sum value of AgentBonus for each category for Column are : Channel

Online	6.816274e+05
Third Party Partner	1.217997e+06
Agent	4.642128e+06

Name: AgentBonus_Per_Policy, dtype: float64

Field name is OCCUPATION : and Distinct Count for this Column is 4
Field name is OCCUPATION : and it's distinct categories are ['Salaried' 'Free Lancer' 'Small Business' 'Large Business']

Counts are each categories of this column are :

Free Lancer	2
Large Business	408
Small Business	1918
Salaried	2192

Name: Occupation, dtype: int64

Average AgentBonus for each categories of this Column are: Occupation

Free Lancer	1174.416667
Large Business	1337.526348
Small Business	1426.108568
Salaried	1486.503817

Name: AgentBonus_Per_Policy, dtype: float64

Total sum value of AgentBonus for each category for Column are : Occupation

Free Lancer	2.348833e+03
Large Business	5.457107e+05
Small Business	2.735276e+06
Salaried	3.258416e+06

Name: AgentBonus_Per_Policy, dtype: float64

Field name is EDUCATIONFIELD : and Distinct Count for this Column is 6
Field name is EDUCATIONFIELD : and it's distinct categories are ['Graduate' 'Post Graduate' 'Under Graduate' 'Engineer' 'Diploma' 'MBA']

Counts are each categories of this column are :

MBA	74
Post Graduate	252
Engineer	408
Diploma	496
Under Graduate	1420
Graduate	1870

Name: EducationField, dtype: int64

Average AgentBonus for each categories of this Column are: EducationField

Engineer	1306.026511
Under Graduate	1416.263650
MBA	1432.172297
Diploma	1454.777688
Graduate	1491.099947
Post Graduate	1515.442857

Name: AgentBonus_Per_Policy, dtype: float64

Total sum value of AgentBonus for each category for Column are : EducationField

MBA	1.059808e+05
Post Graduate	3.818916e+05
Engineer	5.328588e+05
Diploma	7.215697e+05
Under Graduate	2.011094e+06
Graduate	2.788357e+06

Name: AgentBonus_Per_Policy, dtype: float64

Field name is GENDER : and Distinct Count for this Column is 2

Field name is GENDER : and it's distinct categories are ['Female' 'Male']

Counts are each categories of this column are :

Female	1832
Male	2688

Name: Gender, dtype: int64

Average AgentBonus for each categories of this Column are: Gender

Female	1432.857442
Male	1457.126990

Name: AgentBonus_Per_Policy, dtype: float64

Total sum value of AgentBonus for each category for Column are : Gender

Female	2.624995e+06
Male	3.916757e+06

Name: AgentBonus_Per_Policy, dtype: float64

Field name is DESIGNATION : and Distinct Count for this Column is 5

Field name is DESIGNATION : and it's distinct categories are ['Manager' 'Executive' 'VP' 'AVP' 'Senior Manager'] And Counts are each categories of this column are :

VP	226
AVP	336
Senior Manager	676
Manager	1620
Executive	1662

Name: Designation, dtype: int64

Average AgentBonus for each categories of this Column are: Designation

Executive	1234.364811
Manager	1405.986770
Senior Manager	1607.296031
AVP	1897.170685
VP	2161.760324

Name: AgentBonus_Per_Policy, dtype: float64

Total sum value of AgentBonus for each category for Column are : Designation

VP	4.885578e+05
AVP	6.374494e+05
Senior Manager	1.086532e+06

Executive 2.051514e+06
Manager 2.277699e+06
Name: AgentBonus_Per_Policy, dtype: float64

Field name is MARITALSTATUS : and Distinct Count for this Column is 4
Field name is MARITALSTATUS : and it's distinct categories are ['Single' 'Divorced' 'Unmarried' 'Married'] And Counts are each categories of this column are :
Unmarried 194
Divorced 804
Single 1254
Married 2268
Name: MaritalStatus, dtype: int64

Average AgentBonus for each categories of this Column are: MaritalStatus
Divorced 1213.452177
Unmarried 1348.409622
Single 1452.517145
Married 1535.753380
Name: AgentBonus_Per_Policy, dtype: float64

Total sum value of AgentBonus for each category for Column are : MaritalStatus
Unmarried 2.615915e+05
Divorced 9.756155e+05
Single 1.821456e+06
Married 3.483089e+06
Name: AgentBonus_Per_Policy, dtype: float64

Field name is ZONE : and Distinct Count for this Column is 4
Field name is ZONE : and it's distinct categories are ['North' 'West' 'East' 'South'] And Counts are each categories of this column are :
South 6
East 64
North 1884
West 2566
Name: Zone, dtype: int64

Average AgentBonus for each categories of this Column are: Zone
South 1337.038889
North 1423.996665
East 1427.593229
West 1465.141959
Name: AgentBonus_Per_Policy, dtype: float64

Total sum value of AgentBonus for each category for Column are : Zone
South 8.022233e+03
East 9.136597e+04
North 2.682810e+06
West 3.759554e+06
Name: AgentBonus_Per_Policy, dtype: float64

Field name is PAYMENTMETHOD : and Distinct Count for this Column is 4
Field name is PAYMENTMETHOD : and it's distinct categories are ['Half Yearly' 'Yearly' 'Quarterly' 'Monthly'] And Counts are each categories of this column are :
Quarterly 76
Monthly 354
Yearly 1434
Half Yearly 2656
Name: PaymentMethod, dtype: int64

Average AgentBonus for each categories of this Column are: PaymentMethod

Quarterly	1148.781579
Yearly	1438.049361
Monthly	1443.254661
Half Yearly	1461.359130

Name: AgentBonus_Per_Policy, dtype: float64

Total sum value of AgentBonus for each category for Column are : PaymentMethod

Quarterly	8.730740e+04
Monthly	5.109121e+05
Yearly	2.062163e+06
Half Yearly	3.881370e+06

Name: AgentBonus_Per_Policy, dtype: float64

Insights:

Based on the Above counts and unique values of Different Categorical fields of the data, we can come to this conclusion:

- Field name is CHANNEL : and Distinct Count for this Column is 3
 - and it's distinct categories are ['Agent' 'Third Party Partner' 'Online']
 - There are maximum no of insurance taken from Agents with no of 3194 , and least no of insurance taken by online channel with count of 468.
 - On an average all three type of channels receiving about similar bonus, there is very slight difference in Bonus amount, if channel is different.
 - maximum total Bonus received by Agents Categories and Online policy giving agents get least amount of Bonus.
- Field name is OCCUPATION : and Distinct Count for this Column is 4
 - it's distinct categories are ['Salaried' 'Free Lancer' 'Small Business' 'Large Business']
 - Salaried person are among highest Insurance holder with count of 2192, whereas Free lancer taken very less insurance of only 2 records in sample data
 - Again, on an average Bonus remain about same for all agents, who sold policies to any occupation of customer, though Bonus is highest for Salaries Customer.
 - Undouble Total bonus for Agents, who sold policy to Salaried Customers are maximum, because Salaried person are maximum in numbers as well.
- Field name is EDUCATIONFIELD : and Distinct Count for this Column is 6
 - and it's distinct categories are ['Graduate' 'Post Graduate' 'Under Graduate' 'Engineer' 'Diploma' 'MBA']
 - there are maximum Customers are Graduate, whereas MBA holder are least.
- Field name is GENDER : and Distinct Count for this Column is 2
 - and it's distinct categories are ['Female' 'Male']
 - Maximum Customers are Male
- Field name is DESIGNATION : and Distinct Count for this Column is 5
 - and it's distinct categories are ['Manager' 'Executive' 'VP' 'AVP' 'Senior Manager']
 - Counts are each categories of this column are :
 - maximum customers are Executives and least customers are on VP post, this is obvious data.
- Maximum Average bonus given to Agents, who sold policies to VP, it seems, VP takes most Sum insured, thats why Bonus is high for them.

7. Field name is MARITALSTATUS : and Distinct Count for this Column is 4
 - and it's distinct categories are ['Single' 'Divorced' 'Unmarried' 'Married']
 - Counts are each categories of this column are :
 - Married person bought maximum policies.
8. Field name is ZONE : and Distinct Count for this Column is 4
 - and it's distinct categories are ['North' 'West' 'East' 'South']
 - Counts are each categories of this column are :
 - maximum customers belong to West region of Country with the count of 2566, whereas there are only 6 customers from South region, this is interesting data to analyze, why we have very less customers from south. Also customers from east region are also very less with no of 64.
9. Field name is PAYMENTMETHOD : and Distinct Count for this Column is 4
 - and it's distinct categories are ['Half Yearly' 'Yearly' 'Quarterly' 'Monthly']
 - Counts are each categories of this column are :
 - Customers like to give EMI for policy maximumm is half yearly and then yearly.

3.8 Multivariate Analysis

Heat Map:



Figure 4 heat Map

Co-relation Matrix:

	Age	CustTenure	ExistingProdType	NumberOfPolicy	MonthlyIncome	ExistingPolicyTenure	AgentBonus_Per_Policy	SumAssu
Age	1.000000	0.359631	0.076609	0.053143	0.347942	0.263640	0.211933	
CustTenure	0.359631	1.000000	0.115561	0.066859	0.368522	0.347473	0.207415	
ExistingProdType	0.076609	0.115561	1.000000	0.149862	0.191194	0.009800	-0.085003	
NumberOfPolicy	0.053143	0.066859	0.149862	1.000000	0.128328	0.021839	-0.733911	
MonthlyIncome	0.347942	0.368522	0.191194	0.128328	1.000000	0.410641	0.163630	
ExistingPolicyTenure	0.263640	0.347473	0.009800	0.021839	0.410641	1.000000	0.188274	
AgentBonus_Per_Policy	0.211933	0.207415	-0.085003	-0.733911	0.163630	0.188274	1.000000	
SumAssured_Per_Policy	0.197145	0.193545	-0.072151	-0.683045	0.154269	0.185564	0.937796	

Insights:

1. AgentBonus_Per_Policy have very strong relationship with about all of the other fields
2. As Sum insured of the Customer increases, Bonus also increases.
3. Bonus increased with the experience of the Age of the Customer, Customer tenure, Monthly income of the Customer, as well as Existing policy tenure,
4. There is very minute positive relationship between Agent Bonus with Existing Prod type, Number of Policies, and Customer care Score.
5. CustCareScore, LastMonthCalls and Complaint dont have any relation with any other field, its almost 0 co-relation with every other field. So we have dropped them in initial steps

Pair plot:

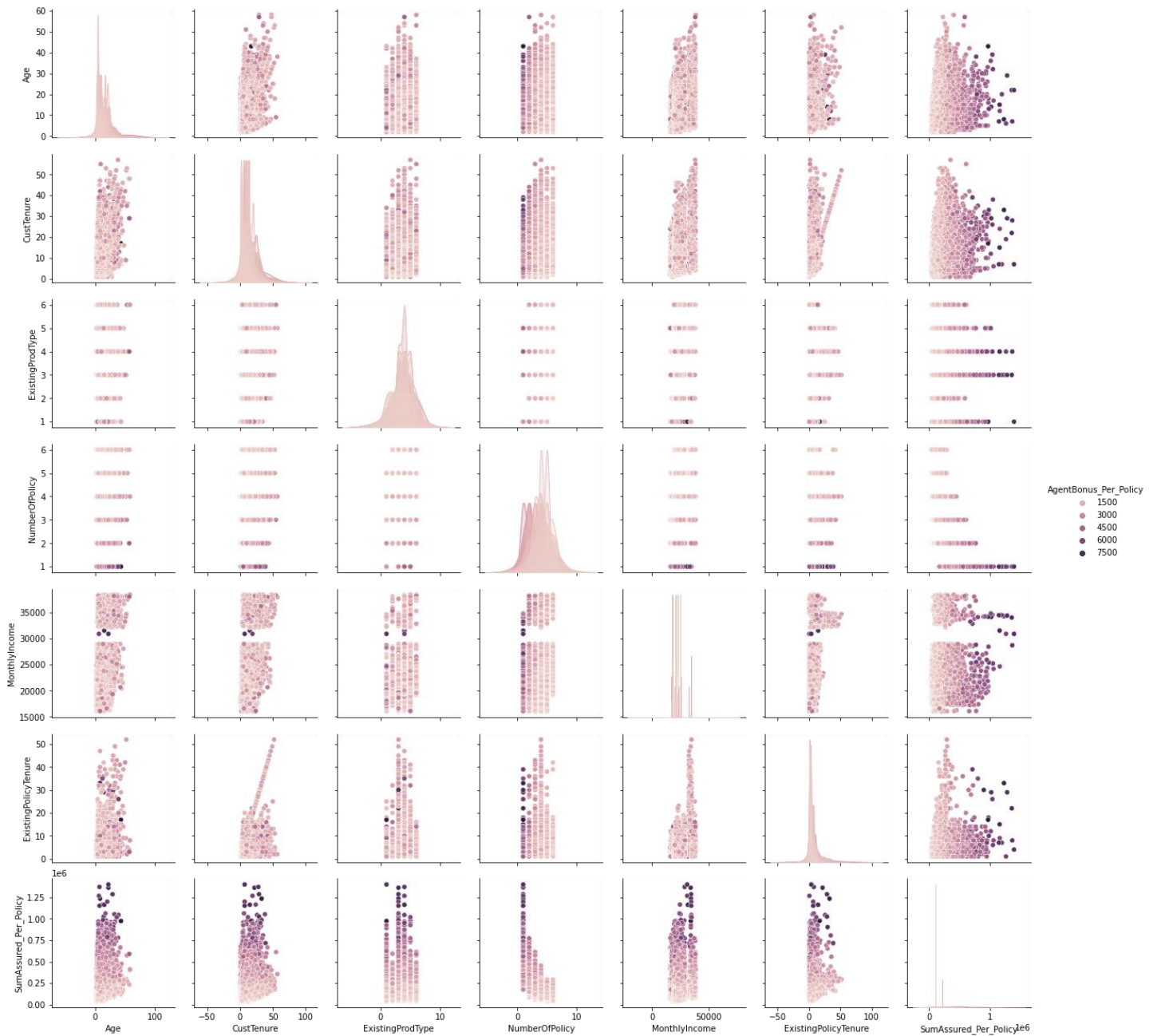


Figure 5 pair plot against AgentBonus per policy

insights:

1. there is clear separation of Bonus for Agents, as all variables increasing, Bonus also increasing.

4. Variable transformation / Data Transformations and Scaling

We have seen the scale from data description, that data have different kind of Scales for AgentBonus, SumAssured Vs Age, CustomerTenure , Last month calls, Complaints and etc. So in order to build a Linear Regression Model, we can Scale the variables using different methods available .

Variable transformation is a way to make the data work better in your model. Data variables can have two types of form: numeric variable and categorical variable, and their transformation should have different approaches.

4.1 Numeric Variable Transformation:

Numeric Variable Transformation is turning a numeric variable to another numeric variable. Typically it is meant to change the scale of values and/or to adjust the skewed data distribution to Gaussian-like distribution through some “monotonic transformation”.

Decision tree-based models are less sensitive to scale and skew than these techniques may not contribute a lot, but for other models (e.g. neural net, SVM, linear model etc.), they could be a game changer, or for some cases even mandatory, such as the case you use penalization terms with L1/L2 norm.

There are many ways, we can achieve Numeric variable transformation or Scale them :

Min-max scaling
Logarithmic transformation
Standardization
RankGauss etc

In our Model , we have built 2 separate data set:

df_scaled for Transformed numeric fields

df_new for another data set for Non transformation of existing Numeric fields, basically use as is.

I have used z-score from Sklearn libraries for transforming all Numeric fields. And once this is done, we will use it in Linear transformation, and we will check , if transformation is reducing noise from the Model or not.

4.2 Categorical Variable Transformation (Encoding)

Categorical variable transformation is turning a categorical variable to a numeric variable. Categorical variable transformation is mandatory for most of the machine learning models because they can handle only numeric values. It is also called 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
```

5. 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	ExistingPolicyTenure	Zone	PaymentMethod	AgentBonus_Per_Policy
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				

6. Model building and interpretation :

6.1 SKLearn Linear Model:

6.1.1 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: $y=ax+b$

6.1.2 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
```

6.1.3 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 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.

6.1.4 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:

$$\text{RMSE} = \text{Square root of } \left(\frac{\sum (P_i - O_i)^2}{n} \right)$$

where:

Σ is a fancy symbol that means “sum”

P_i is the predicted value for the i th observation in the dataset

O_i is the observed value for the i th 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:

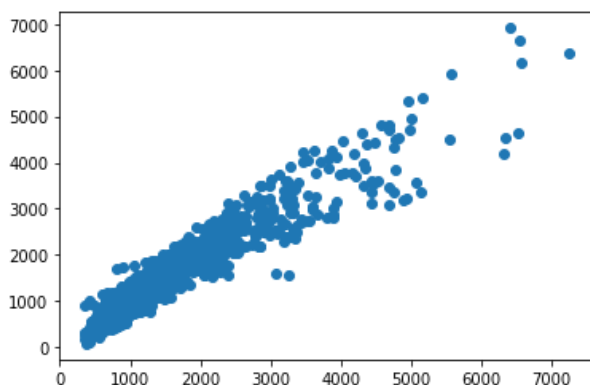


Figure 6 Scatter plot for Prediction on Test data

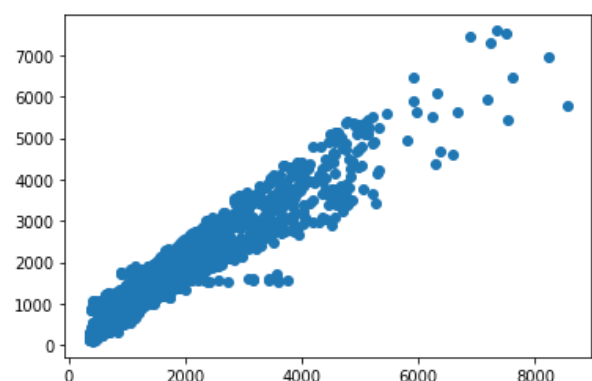


Figure 7 Scatter plot for Prediction on Train data

6.1.5 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
```

6.1.6 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.

6.2 Building Linear Model with Scaled parameters :

6.2.1 Definition:

We have Scaled numeric fields from Sklearn's z-score library and built new data frame `df_scaled`. We used this new data frame for building Linear regression model and will check, how the model performs.

Decision tree-based models are less sensitive to scale and skew than these techniques may not contribute a lot, but for other models (e.g. neural net, SVM, linear model etc.), they could be a game changer, or for some cases even mandatory, such as the case you use penalization terms with L1/L2 norm.

6.2.2 Action:

We have performed linear regression techniques from Sklearn LinearRegression library .

Following is co-efficient of Determination :

The coefficient for Age is 42.83962224334871
The coefficient for CustTenure is 43.253731727586946
The coefficient for Channel is -7.047067651864262
The coefficient for Occupation is 4.781586127409858
The coefficient for EducationField is -9.566226806308082
The coefficient for Gender is 1.047841452593378
The coefficient for ExistingProdType is -15.512531622244383
The coefficient for Designation is 4.601985548114451
The coefficient for NumberOfPolicy is -231.44126984310716
The coefficient for MaritalStatus is 11.3843211448863
The coefficient for MonthlyIncome is 41.482235158809694
The coefficient for ExistingPolicyTenure is 13.43519903309641
The coefficient for Zone is -5.700509672207306
The coefficient for PaymentMethod is 2.205000719314816
The coefficient for SumAssured_Per_Policy is 772.295331497455

6.2.3 Equation:

$$\begin{aligned} &\text{Age} * 42.83962224334871 + \text{CustTenure} * 43.253731727586946 + \text{Channel} * -7.047067651864 \\ &262 + \text{Occupation} * 4.781586127409858 + \text{EducationField} * -9.566226806308082 + \text{Gender} * \\ &1.047841452593378 + \text{ExistingProdType} * -15.512531622244383 + \text{Designation} * 4.60198554 \\ &8114451 + \text{NumberOfPolicy} * -231.44126984310716 + \text{MaritalStatus} * 11.3843211448863 + \text{M} \\ &\text{onthlyIncome} * 41.482235158809694 + \text{ExistingPolicyTenure} * 13.43519903309641 + \text{Zone} * \\ &-5.700509672207306 + \text{PaymentMethod} * 2.205000719314816 + \text{SumAssured_Per_Policy} * 772. \\ &295331497455 + 1447.1195249943657 \end{aligned}$$

6.2.4 Performance parameters:

The coefficient of determination R^2 of the prediction on Train set 0.9067763446568196

The coefficient of determination R^2 of the prediction on Test set 0.8964948926865648

The intercept for the model is 1447.1195249943657

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 Root Mean Square Error (RMSE) of the model is for Train data set is 317.48346002535106

The Root Mean Square Error (RMSE) of the model is for testing set is 316.63499771886006

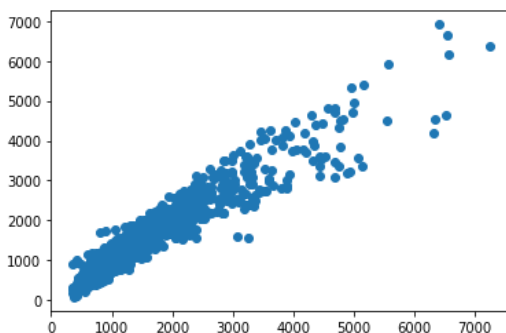


Figure 8 Scatter plot for Prediction on Scaled Test data

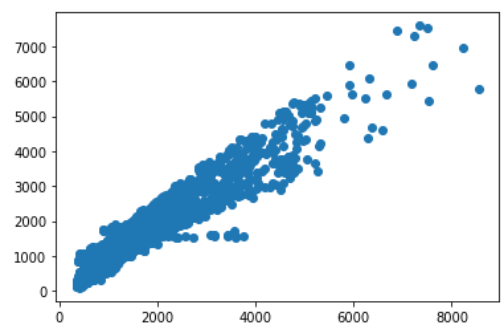


Figure 9 Scatter plot for Prediction on Scaled train data

Train data matrix:

```
The MAE is: 220.32100694973744
The MSE is: 100795.74738966869
The MAPE is 0.17072418037188575
The EVS is 0.9067763446568196
The RMSE 317.48346002535106
```

Test data matrix:

```
The MAE for Test data is: 222.51949001959946
The MSE for Test data is: 100257.72178042251
The MAPE for Test data is 0.17126532112977647
The EVS for Test data is 0.8964952272518005
The RMSE 316.63499771886006
```

6.3 Building Stats Model:

6.3.1 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.

6.3.2 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

6.3.3 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

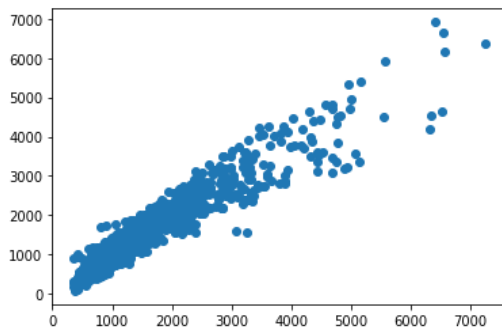


Figure 10 Scatter Plot for Predicted values on Test data set

6.3.4 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.

6.4 Building Multiple Stats Model to eliminate non affecting fields :

6.4.1 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.

6.4.1.1 How to interpret a given VIF value?:

Consider the following linear regression model: $Y = \beta_0 + \beta_1 \times X_1 + \beta_2 \times X_2 + \beta_3 \times X_3 + \epsilon$

For each of the independent variables X_1 , X_2 and X_3 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 X_1 : $VIF_1 = 1 / (1 - R^{*2})$

VIF (variance inflating factor) formula for the first variable in the model.

R^2 in this formula is the coefficient of determination from the linear regression model which has:

X_1 as dependent variable

X_2 and X_3 as independent variables

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

$$X1 = \beta_0 + \beta_1 \times X2 + \beta_2 \times X3 + \epsilon$$

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.

6.4.1.2 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

6.4.1.3 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

6.4.1.4 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

6.4.1.5 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:

- 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 .
- 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 .
- 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 .
- 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 .
- 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 .
- 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 .
- 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.

6.4.1.6 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

6.4.1.7 Equation and interpretation:

We can use below formula for determining the targeted value:

$$\text{AgentBonus_per_policy} = (121.59) + (2.53) * \text{CustTenure} + 2.28) * \text{ExistingPolicyTenure} + (0.01) * \text{SumAssured_Per_Policy}$$

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

6.5 Decision Tree Model

6.5.1 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:

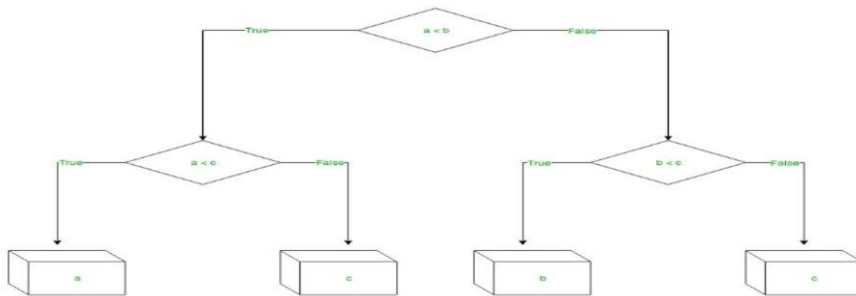


Figure 11 Example for Decision tree

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.

6.5.2 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 .

6.5.3 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

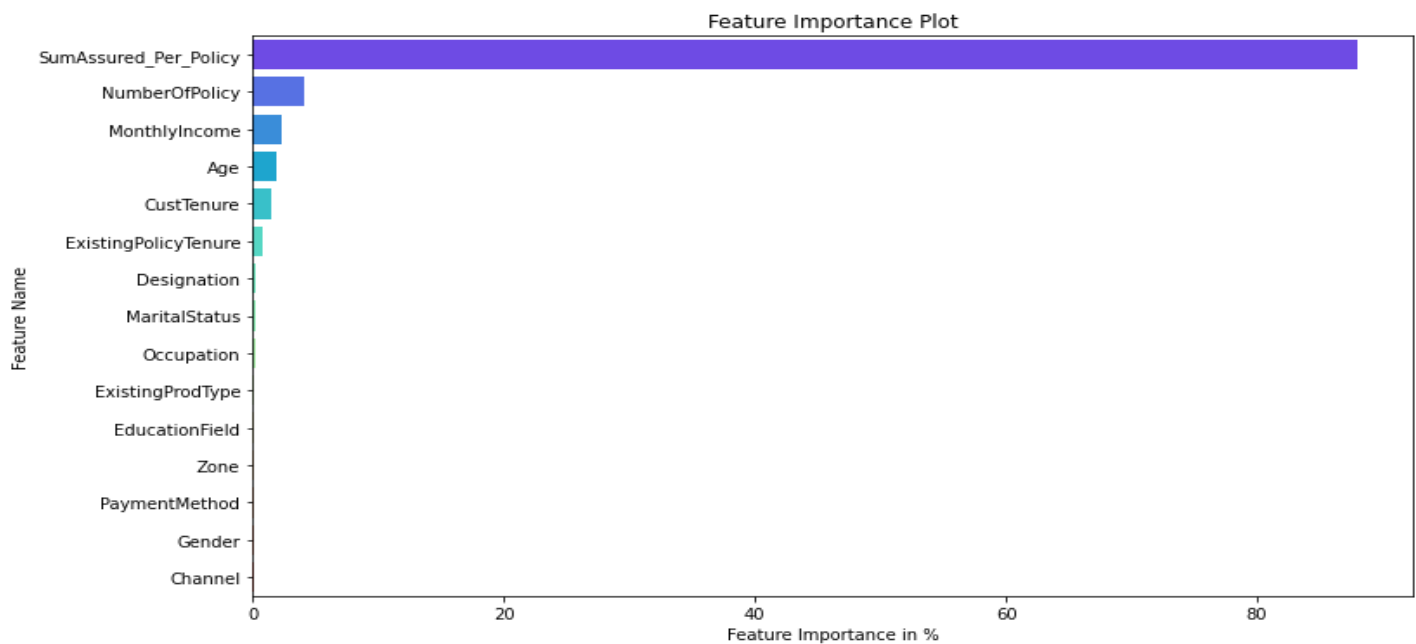


Figure 12 Decision Tree Feature Importance

6.5.4 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

6.6 Random Forest Regressor Model

6.6.1 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**.

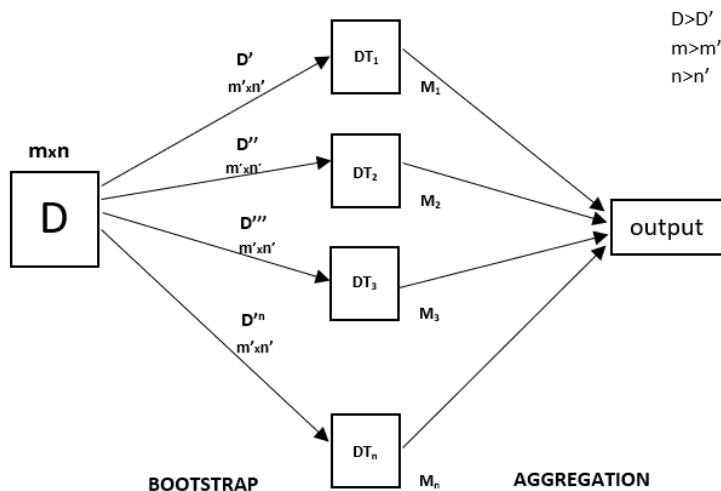


Figure 13 Example for Random Forest

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

6.6.2 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 .

6.6.3 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

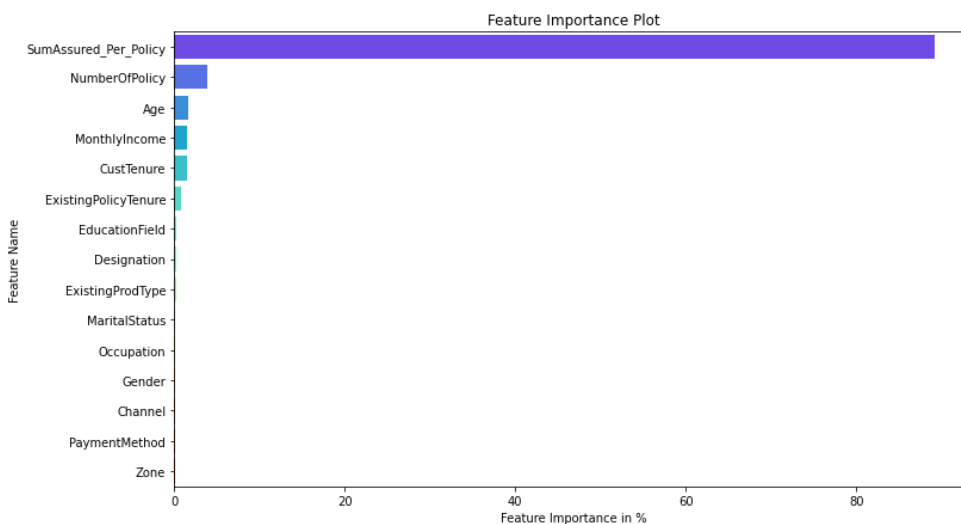


Figure 14 Random Forest Feature Importance

6.6.4 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

6.7 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.

6.7.1 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.

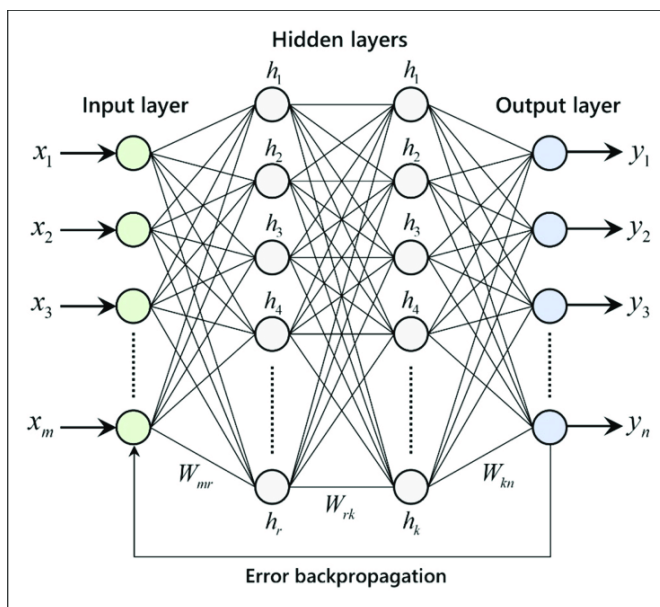


Figure 15 Example for ANN Model

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

6.7.2 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/Sigmoid 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.

6.7.3 Feature Importance :

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

6.7.4 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

7. 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 :

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

7.1.1 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.

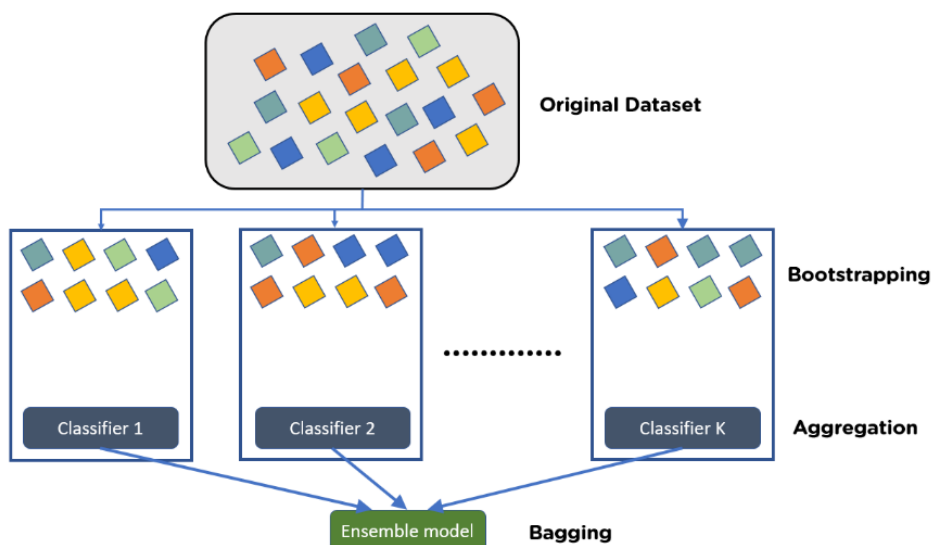


Figure 16 Explain Bagging

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

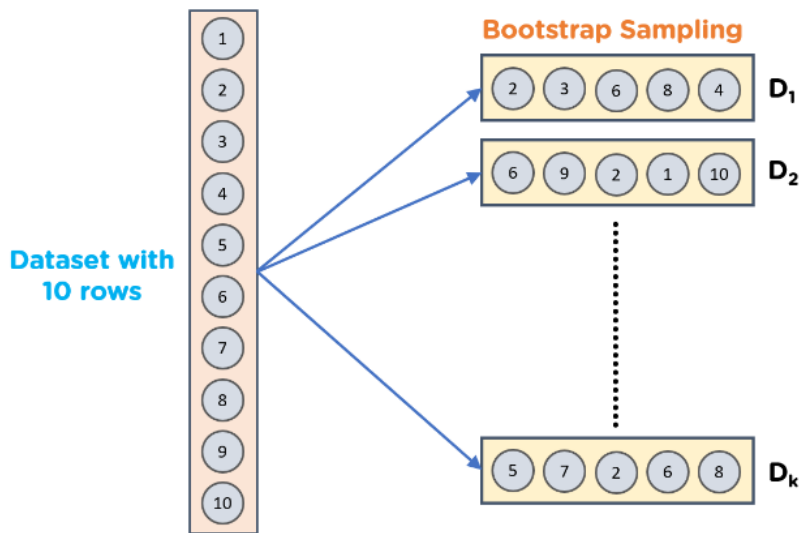


Figure 17 Explain Bootstrapping

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

7.1.2 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.

7.1.3 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

7.1.4 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.

7.2 Hyper parameter tuning for random forest

7.2.1 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 $\text{max_features} = \sqrt{n_features}$
 - In case of sqrt: considers $\text{max_features} = \sqrt{n_features}$, it is same as auto
 - In case of log2: considers $\text{max_features} = \log_2(n_features)$
 - In case of None: considers $\text{max_features} = n_features$
- **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.

7.2.2 Action:

We have used following parameters and tested in gridSearch algorithm .

```
param_grid = { 'max_depth': [8,10,12],  
              'max_features': [3,4,5],  
              '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}
```

7.2.3 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

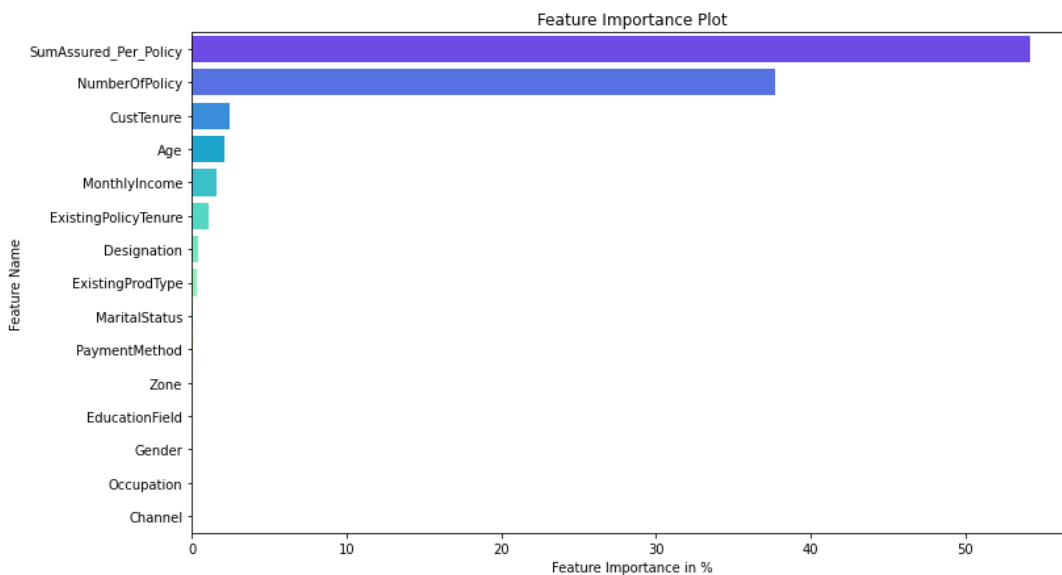


Figure 18 Best Param Random Forest Feature Importance

7.2.4 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.

7.3 Hyper parameter tuning for Decision Tree Regressor

7.3.1 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.

7.3.2 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)
```

7.3.3 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

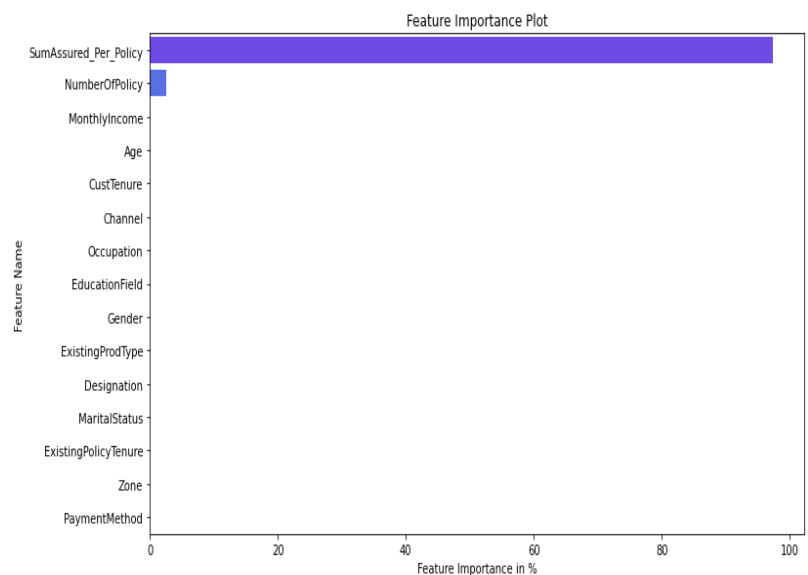


Figure 19 Best feature Decision Tree Tunned Param

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

7.3.4 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 .

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

8.1 Performance Parameters List :

We have created multiple models, and Tunned them as well by gridSearch mechanism. We have considered these performance 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 absolute error of 90 lbs – 89 lbs = 1 lbs. And Mean Absolute Error(MAE) is the average of all absolute errors. The formula is:

$$MAE = \frac{1}{n} \sum_{i=1}^n |x_i - x|$$

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 difference between the observed and predicted values. When a model has no error, the MSE equals zero. As model error increases, 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 percentage 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

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

Where:

n is the number of fitted points,

A_t is the actual value,

F_t 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 the 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:

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

Where:

f = forecasts (expected values or unknown results),

o = observed values (known results).

8.2 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 (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	0.8964
Linear Reg Scaled Data	317.48346	316.634998	220.321007	222.51949	100795.74739	100257.72178	0.170724	0.171265	0.906776	0.8964
Stats Model 1	317.48346002535055	316.63499771886285	220.321007	222.51949	100795.74739	100257.72178	0.170724	0.171265	0.906776	0.8964
Stats Model 2	354.54396801961036	353.56277958413546	238.541875	244.141863	125701.425259	125006.639107	0.175574	0.179678	0.883742	0.8709
Stats Model 3	354.5472293235076	353.5568690941128	238.571656	244.150994	125703.737821	125002.459684	0.175633	0.179681	0.88374	0.870
Stats Model 4	354.57408724890837	353.35723000116565	238.559194	244.031095	125722.783348	124861.331994	0.175619	0.17965	0.883722	0.8711
Stats Model 5	354.6259691021416	353.20780003269664	238.722552	243.744927	125759.577962	124755.750004	0.175782	0.179249	0.883688	0.8712
Stats Model 6	354.6848065841537	353.0174978354055	238.752834	243.6696	125801.312022	124621.353778	0.17576	0.179242	0.883649	0.8713
Stats Model 7	354.7636336409267	352.8757950057764	238.73769	243.744927	125857.235754	124521.326701	0.175744	0.179435	0.883598	0.8714
Stats Model 8	354.8816410065518	352.68578464437667	238.760798	243.458747	125940.979124	124387.26269	0.175876	0.179265	0.88352	0.8716
Stats Model 9	355.0675639410347	352.60212402026525	238.687196	243.298407	126072.974963	124328.257864	0.175543	0.178756	0.883398	0.8716
Decision Tree	0.0	374.135915	0.0	242.624459	0.0	139977.682604	0.0	0.174629	1.0	0.8559
Random Forest	101.955542	270.907949	65.695396	178.33923	10394.932541	73391.11693	0.047686	0.128212	0.990386	0.9242
ANN Model	5082.681797	5093.46129	4972.297786	4990.048298	25833654.247346	25943347.91466	5.255689	5.291988	-0.092953	-0.0999
Random Forest Bagging Model	157.545892	269.293587	106.47556	180.502449	24820.707968	72519.036197	0.076903	0.129404	0.977046	0.9251
Random Forest Tunned Param	157.545892	269.293587	106.47556	180.502449	24820.707968	72519.036197	0.076903	0.129404	0.977046	0.9251
decision_tree tunned param	331.648706	330.099188	199.284134	211.171813	109990.863964	108965.474203	0.147677	0.156464	0.898272	0.8875

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 Reg Scaled Data	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
Stats Model 1	317.483460	316.634998	220.321007	222.51949	100795.74739	100257.72178	0.170724	0.171265	0.906776	0.896495
decision_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	374.135915	0.0	242.624459	0.0	139977.682604	0.0	0.174629	1.0	0.855904
ANN Model	5082.681797	5093.461290	4972.297786	4990.048298	25833654.247346	25943347.91466	5.255689	5.291988	-0.092953	-0.099901

Figure 20 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
decision_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 Reg Scaled Data	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	374.135915	0.0	242.624459	0.0	139977.682604	0.0	0.174629	1.0	0.855904
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 21 Sort 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 , Linear Regression with scaled data 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.

9. Business insights , Recommendations

9.1 Is the data unbalanced? If so, what can be done?:

Since this requirement is for predicting continuous variable and its not a classification model, where we have separate classes for 1s and 0s, which are Targeted values, so we can't say that data is unbalanced.

Also, we have given data set with Agent Bonus for Total SumAssured for N number of policies purchases by any csutomers, so for building good prediction model, we have created 2 new fields , which are

AgentBonus_per_policy

SumAssured_per_Policy

And we have used NumberOfPolicy field for generating above fields.

Some of requirements of doing this are as follows:

- This is very important to know what is Bonus given for each Customers, irrespective of how many policies Customer purchased.
- We need to know, how is SumAssured affected AgenBonus, but if we not created new field SumAssured_per_policy (This is actually a average of Total SumAssured / Number of Policy), Our model wont be able to understand, this relations.
- For an example , if Customer purchased 4 policy of total 1000 Rs, and AgentBonus is 400 Rs, which means Bonus is $1000/(400*4) = 6.25\%$ per policy Bonus
And for any other Customer purchased Single Policy of amount 1000 and his Agent get 250 Rs Bonus, which means, he received $1000/250 = 40\%$ Bonus,
so for making them all on same Scale, we need to create above listed both variable to know Average of AgentBonus per Policy as well as Average of SumAssured per policy.
- We have also checked for the co-relation between all the fields and targeted field, which is AgentBonus and we found that their co-relation coefficient was very low and we dropped those fields.
Complaint : 0.025091
LastMonthCalls: 0.038717
CustCareScore: -0.005319

we have checked that following fields have very minute impact on targeted fields and these can be dropped as well:

9.2 Business insights using clustering

Though this prediction Model don't need any Clustering methodology, but we can build Clusters based on “**Number of Policy**” purchased by any Customers and build separate model for each data set.

And counts in each Clusters would be like this :

NumberOfPolicy	Counts
1.0	438
2.0	711
3.0	939
4.0	1139
5.0	856
6.0	437

Clustering can be also performed using K-mean Clusters, and we can have N number of Clusters, needed. But for this model we don't need this.

Though based on Bivariate analysis, if we are supposed to build any Clusters based on our categorical fiels, we can build Clusters against these fields:

Based on Columns: Channel
Cluster names would be : ['Agent', 'Third Party Partner', 'Online']

Based on Columns: Occupation
Cluster names would be :
['Salaried', 'Free Lancer', 'Small Business', 'Large Business']

Based on Columns : EducationField
Cluster names would be :
['Graduate', 'Post Graduate', 'Under Graduate', 'Engineer', 'Diploma', 'MBA']

Based on Columns: Gender

Cluster names would be : ['Female', 'Male']

Based on Columns: Designation

Cluster names would be : ['Manager', 'Executive', 'VP', 'AVP', 'Senior Manager']

Based on Columns: MaritalStatus

Cluster names would be : ['Single', 'Divorced', 'Unmarried', 'Married']

Based on Columns: Zone

Cluster names would be : ['North', 'West', 'East', 'South']

Based on Columns: PaymentMethod

Cluster names would be : ['Half Yearly', 'Yearly', 'Quarterly', 'Monthly']

9.3 Business insights from EDA

We performed following actions on our data :

- Created 2 new fields AgentBonus_per_policy and SumAssured_per_policy, which is basically Average AgentBonus and Average SumAssured
- We dropped fields CustID, AgentBonus and SumAssured for building our model.
- Model building will be done in future submissions.
- We have imputed NULL values for each of our missing field values.
- I have analyzed data for Univariate analysis, checked individual column's distribution, their counts, in it's category, mean and Sum of values. Checked outliers, distribution and checked normality of data.
- I have also done Co-relation check, data skewness and checked distribution of all the fields against target variable.

Based on above analysis , following are business insights:

- We can see clear co-relation between SumAssured_per_policy and AgentBonus_per_policy field with co-relation of 0.94 , which means, as Sumassured increases, AgentBonus also increases.
- AgentBonus also have good positive co-relation with field Age, CustomerTenure and existingPolicyTenure of with Positive 0.2 with each field mentioned.
- AgentBonus have very minute relation with fields Complaint and LastMonthCalls, which makes sense.
- There are very less policies sold in South region, and we have only 6 customers from South region, insurance company needs to give good campaign in that region, and there is good amount of chance to spread into different geolocation. And spread their legs in that.
- Also, Company should run motivational trainings, good sales skill development program to their South region Agents.

9.4 Business benefits:

- Company would know strength of their own employees.
- They can easily segregate good Salesperson and low performing Agents.
- Company can give clear criteria, by informing Salesperson, how they can achieve good incentives.
- It will remove partiality and bad judgements of the top-level managements
- It will create good environment among all employees for fare justifications of the Bonus amounts.
- Company can encourage good performing Sales-persons
- Company can also provide Trainings or build any special program to assist bad Agents.
- Additional steps can be taken by Company by accompanying good Agent with low performing Agents, so that they can learn from good resources.

9.5 Improvement Area and recommendations:

- South Zone have least number of Customers, so good salesperson should deploy to tough market
- There can be some discount offer or Less policy premium campaign can be run in South and East regions, where policy Holders are very less.
- There is good opportunity to attract more Large business holders, they have deep pockets and Company can make good money from them.
- Maximum number of policies sold by Agents and very less policies sold by Online channel. So we can provide some sort of discounts and promotions for Online Agents, in order for making good online business, Online channels are very powerful nowadays and it has reach to every single customers, so make it powerful.
- Maximum number of policy holders are Salaried with count of 2192 records and least number of Policy holders are free Lancer. Insurance company can reach to more and more Free Lancing Customers, as they are very less from the Sample customers.
- Maximum number of policy holders are Graduate in Education with count of 1870 records, and MBA holders are least with no of 74. Insurance company can reach to more and more MBA professionals as they very less from the Sample customers.
- Insurance company can reach to more female customers for buying policies and do not and give some promotions to women. As Male are Maximum number of policy holders as compared to female records.
- Married people are Maximum number of policy holders whereas unmarried people don't prefer taking Insurance policy. So Company can run campaign and encourage more youths for knowing about policies and how important is insurance nowadays .