CAPSTONE PROJECT-

LIFE INSURANCE SALES

Created by-

SHUBHASREE SARKAR

Ackowledgement

The success and final outcome of this capstone project required a lot of guidance and assistance from many people and I am extremely privileged to have got this all along the completion of my project. All that I have done is only due to such supervision and assistance and I would not forget to thank them.

I would like to thank Great Learning team for providing me an opportunity to do this project work and giving us all support and guidance, which made me complete the project duly.

I am extremely thankful to our trainers Mr. Mohamad Dawood Farzan Sajahan and Mr. Animesh Tiwari, for providing such a nice support and guidance from the start of this course and throughout the whole process.

I owe my deep gratitude to our project guide Mr. Aniket Chhabra, who took keen interest on our project work and guided us all along, till the completion of our project work by providing all the necessary information for developing a good system and providing us invaluable feedback, although he had busy schedule managing the corporate affairs.

I would not forget to remember Ms. Anjali Grover, of Great Learning for her constant encouragement and more over for her timely support and guidance till the completion of our project work.

Lastly, I would like to thank my batchmates and everyone else, with whom I had the opportunity to learn together and could learn from. I wish you all the best in your life and career.

Thank you,

Shubhasree Sarkar

INDEX

Sr No	Topics	Page No
1	**Introduction**	**5**
	- Brief introduction about the problem statement and the need of solving it.	
2	**EDA and Business Implication**	**6-24**
	- Uni-variate / Bi-variate / Multi-variate analysis to understand relationship b/w variables. How your analysis is impacting the business?	
	- Both visual and non-visual understanding of the data.	
3	**Data Cleaning and Pre-processing**	**25-27**
	- Approach used for identifying and treating missing values and outlier treatment (and why)	25-26
	- Need for variable transformation (if any)	27
	- Variables removed or added and why (if any)	27
4	**Model building**	**28-114**
	- Clear on why was a particular model(s) chosen.	28-96
	- Effort to improve model performance.	97-114
5	**Model validation**	**115**
	- How was the model validated? Just accuracy, or anything else too?	
6	**Final interpretation / recommendation**	**116-117**
	- Detailed recommendations for the management/client based on the analysis done.	
	Reference	**118-119**

List of Tables

Description	Page no.
Dataset Overview	**6**
The Data Description	**7**
The Summary Of The Continuous Variables- Before converting to categorical	**7**
The Summary Of The Continuous Variables- After converting to categorical	**7**
The Summary Of The Categorical Variables	**10**
The variable info	**13**
The Datatypes	**13**
Missing Values	**25**
X_Train datatset-Linear Regression(After Scaling)-Table	**42**
X_Test datatset- Linear Regression (After Scaling)-Table	**42**
Coefficients of different features - Table	**43**
Metrices in Linear Regression-Table	**44**
Stats model Summary Output Table	**45**
VIF Values in Linear Regression- Table	**46**
X_Train datatset-Ridge Regression (After Scaling)-Table	**52**
X_Test datatset-Ridge Regression (After Scaling)-Table	**52**
Coefficients of different features - Table	**53**
Metrices in Ridge Regression-Table	**54**
X_Train datatset-Lasso Regression (After Scaling)-Table	**59**
X_Test datatset-Lasso Regression (After Scaling)-Table	**59**
Coefficients of different features - Table	**60**
Metrices in Lasso Regression-Table	**61**
Actual vs Predicted Values- Decision Tree Regressor	**66**
Metrices in Decision Tree Regressor-Table	**67**
Actual vs Predicted Values- Random Forest Regressor	**73**
Metrices in Random Forest Regressor-Table	**74**
X_Train datatset-KNN(After Scaling)-Table	**79**
X_Test datatset-KNN(After Scaling)-Table	**79**
Actual vs Predicted Values-KNN	**80**
Metrices in KNN Regressor-Table	**81**
Actual vs Predicted Values- Gradient Boost Regressor	**87**
Metrices in Gradient Boost Regressor-Table	**88**
Actual vs Predicted Values- XG Boost Regressor	**93**
Metrices in XG Boost Regressor-Table	**94**
Actual vs Predicted Values-Decision Tree Regressor-After using Gridsearch	**99**
Metrices in Decision Tree Regressor (Using Grid Search)-Table	**100**
Actual vs Predicted Values-Random Forest Regressor-After using Gridsearch	**102**
Metrices in Random Forest Regressor(Using Grid Search)-Table	**103**
Actual vs Predicted Values-KNN-After using Gridsearch	**105**
Metrices in KNN Regressor (Using Grid Search)-Table	**106**
Actual vs Predicted Values-Gradient Boosting Regressor-After using Gridsearch	**108**
Metrices in Gradient Boosting Regressor(Using Grid Search)-Table	**109**
Actual vs Predicted Values-XG Boosting Regressor-After using Gridsearch	**111**
Metrices in XG Boosting Regressor (Using Grid Search)-Table	**112**

List of Figures

Description	Page no.
Distribution plots (histogram)	**14**
Box plots (without outlier treatment) for the continuous columns	**16**
Skewness of the Continuous variables	**17**
Bar plots for the categorical columns	**18**
Pair plots for Bi variate data	**20**
Heat map with only continuous variables	**21**
Bar plots for the categorical columns w.r.t. AgentBonus	**23**
Box plots (with outlier treatment) for the continuous columns	**26**
Variable Transformation - AgentBonus	**27**
Scatter plot of Actual vs Predicted price- Linear Regression	**47**
Scatter plot of Actual vs Predicted price- Ridge Regression	**55**
Scatter plot of Actual vs Predicted price- Lasso Regression	**62**
Scatter plot of Actual vs Predicted price- Decision Tree Regressor	**68**
Scatter plot of Actual vs Predicted price- Random Forest Regressor	**75**
Scatter plot of Actual vs Predicted price-KNN	**82**
Scatter plot of Actual vs Predicted price- Gradient Boost Regressor	**89**
Scatter plot of Actual vs Predicted price- XG Boost Regressor	**95**
Scatter plot of Actual vs Predicted price-Decision Tree Regressor- Using Grid Search	**101**
Scatter plot of Actual vs Predicted price- Random Forest Regressor - Using Grid Search	**104**
Scatter plot of Actual vs Predicted price-KNN- Using Grid Search	**107**
Scatter plot of Actual vs Predicted price- Gradient Boosting Regressor- Using Grid Search	**110**
Scatter plot of Actual vs Predicted price-XG Boosting Regressor- Using Grid Search	**113**
Model Validation Table	**115**

Topic: Life Insurance Sales

1) Introduction

Brief introduction about the problem statement and the need of solving it.

a) 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.

So basically, the company wants to build an engagement strategy using the model which will engage the company's employees or agents. They want to assign more bonus to agents who are performing well as compared to those performing less using an incentive-based performance mode

So, the target variable is Present month Bonus or CurrentMonthAgentBonus

b) Need of the problem statement

The need of the study or the project is to understand the impact of the Life Insurance Sales amount to the Bonus associated with Individual Agent's role which in turn is a determinant of their performance in the industry. This is also an important project in the BFSI segment, considering how the data is being collected and what variables mostly define the Bonus of an Agent in such an industry.

c) Understanding business/social opportunity

Social opportunity- Training the agents ethically in a better way as life insurance is related to social cause as society can benefit from the insurance policies.

Life insurance is a subject upon which neither prospects nor financial advisors care to dwell. Product designs can be complex, and events that trigger benefits don't evoke pleasant images, unlike retirement planning ads that contain sailboats and strolls on the beach.

However, the need for life insurance exists across a large segment of consumers who generate income and borrow money. Finding ways to increase life insurance sales reverts back to some time-tested methods and involves a few new wrinkles to help advisors tap evolving markets.

- **Prospective buyers of life insurance are reassured when you know your product inside and out.**
- **However, keep your sales presentation simple and brief. Be authentic in your conversation.**
- **Building a referral network can help increase sales, as well as sourcing leads via social media.**

****

2. EDA and Business Implication

- Uni-variate / Bi-variate / Multi-variate analysis to understand relationship b/w variables. How your analysis is impacting the business?
- Both visual and non-visual understanding of the data.

a) Non- Visual inspection of data (rows, columns, descriptive details)

Dataset Overview

Target Variable	Agent Bonus
Defining Agent Bonus	Bonus Amount given to each Agent in the last month. Bonus is to be calculated based on the performance
Time Period	Details can't be said as it is a secondary without datetime attribute
Analysed Population	Sample dataset of 4520 individuals (Attributes(Columns-20), (Rows- 4520))
Modelling Technique	Regression model
Validation Window	Future datasets

The total no of rows and columns are being shown below-

```
The number of rows (observations) is 4520
The number of columns (variables) is 20
```

The Data Description

Data	Variable	Discerption
Sales	CustID	Unique customer ID
Sales	AgentBonus	Bonus amount given to each agents in last month
Sales	Age	Age of customer
Sales	CustTenure	Tenure of customer in organization
Sales	Channel	Channel through which acquisition of customer is done
Sales	Occupation	Occupation of customer
Sales	EducationField	Field of education of customer
Sales	Gender	Gender of customer
Sales	ExistingProdType	Existing product type of customer
Sales	Designation	Designation of customer in their organization
Sales	NumberOfPolicy	Total number of existing policy of a customer
Sales	MaritalStatus	Marital status of customer
Sales	MonthlyIncome	Gross monthly income of customer
Sales	Complaint	Indicator of complaint registered in last one month by customer
Sales	ExistingPolicyTenure	Max tenure in all existing policies of customer
Sales	SumAssured	Max of sum assured in all existing policies of customer
Sales	Zone	Customer belongs to which zone in India. Like East, West, North and South
Sales	PaymentMethod	Frequency of payment selected by customer like Monthly, quarterly, half yearly and yearly
Sales	LastMonthCalls	Total calls attempted by company to a customer for cross sell
Sales	CustCareScore	Customer satisfaction score given by customer in previous service call

The Summary Of The Continuous Variables- Before converting to categorical

	count	mean	std	min	25%	50%	75%	max
CustID	4520.0	7.002260e+06	1304.955938	7000000.0	7001129.75	7002259.5	7003389.25	7004519.0
AgentBonus	4520.0	4.077838e+03	1403.321711	1605.0	3027.75	3911.5	4867.25	9608.0
Age	4251.0	1.449471e+01	9.037629	2.0	7.00	13.0	20.00	58.0
CustTenure	4294.0	1.446903e+01	8.963671	2.0	7.00	13.0	20.00	57.0
ExistingProdType	4520.0	3.688938e+00	1.015769	1.0	3.00	4.0	4.00	6.0
NumberOfPolicy	4475.0	3.565363e+00	1.455926	1.0	2.00	4.0	5.00	6.0
MonthlyIncome	4284.0	2.289031e+04	4885.600757	16009.0	19683.50	21606.0	24725.00	38456.0
Complaint	4520.0	2.871681e-01	0.452491	0.0	0.00	0.0	1.00	1.0
ExistingPolicyTenure	4336.0	4.130074e+00	3.346386	1.0	2.00	3.0	6.00	25.0
SumAssured	4366.0	6.199997e+05	246234.822140	168536.0	439443.25	578976.5	758236.00	1838496.0
LastMonthCalls	4520.0	4.626991e+00	3.620132	0.0	2.00	3.0	8.00	18.0
CustCareScore	4468.0	3.067592e+00	1.382968	1.0	2.00	3.0	4.00	5.0

The Existing Product Type,Number of Policy,Complaint, CustCareScore are Categorical variables, as can be confirmed while looking at the data

- They are discrete in nature

The Summary Of The Continuous Variables- After converting to categorical

	count	mean	std	min	25%	50%	75%	max
CustID	4520.0	7.002260e+06	1304.955938	7000000.0	7001129.75	7002259.5	7003389.25	7004519.0
AgentBonus	4520.0	4.077838e+03	1403.321711	1605.0	3027.75	3911.5	4867.25	9608.0
Age	4251.0	1.449471e+01	9.037629	2.0	7.00	13.0	20.00	58.0
CustTenure	4294.0	1.446903e+01	8.963671	2.0	7.00	13.0	20.00	57.0
MonthlyIncome	4284.0	2.289031e+04	4885.600757	16009.0	19683.50	21606.0	24725.00	38456.0
ExistingPolicyTenure	4336.0	4.130074e+00	3.346386	1.0	2.00	3.0	6.00	25.0
SumAssured	4366.0	6.199997e+05	246234.822140	168536.0	439443.25	578976.5	758236.00	1838496.0
LastMonthCalls	4520.0	4.626991e+00	3.620132	0.0	2.00	3.0	8.00	18.0

Continuous Variables Data Description

The above figure shows the statistical measures associated with the continuous variables.

The description of the same is shown below-

CustID

Here *CustID* is actually *Unique customer ID*
Total count is **4520.**
No presence of **missing values**

AgentBonus

Here AgentBonus is actually *Bonus amount given to each agent in last month*
Average AgentBonus is **4078 units** with **standard deviation** of **1403.11units**
The AgentBonus range is between **1605 units (Minimum)** and **9608 units (Maximum)**
The **median** of the AgentBonus is **3911.5 units**
Total count is **4520.**

Age

Here Age is actually *Age of customer*
Average Age is **14 units** with **standard deviation** of **9.03 units**
The Age range is between **9.03 units (Minimum)** and **58 units (Maximum)**
The **median** of the Age is **13 units**
Total count is **4251(Missing values present)**

CustTenure

Here CustTenure is actually *Tenure of customer in organization*
Average CustTenure is **14.4 units** with **standard deviation** of **8.96 units**
The CustTenure range is between **2.0 units (Minimum)** and **57 units (Maximum)**
The **median** of the CustTenure is **13 units**
Total count is **4294(Missing values present)**

MonthlyIncome

Here MonthlyIncome is actually *Gross monthly income of customer*
Average MonthlyIncome is **22890 units** with **standard deviation** of **4885.6 units**
The MonthlyIncome range is between 16009.0 **units (Minimum)** and 38456.0 **units (Maximum)**
The **median** of the MonthlyIncome is 21606.0 **units**
Total count is **4284**

ExistingPolicyTenure

Here ExistingPolicyTenure is actually *Max tenure in all existing policies of customer.*
Average ExistingPolicyTenure is **4.13 units** with **standard deviation** of **3.34 units**
The ExistingPolicyTenure range is between 1 **units (Minimum)** and **25 units (Maximum)**
The **median** of the ExistingPolicyTenure is **3 units**
Total count is **4336.**

SumAssured

Here SumAssured is actually *Max tenure in all existing policies of customer.*
Average SumAssured is **619997 units** with **standard deviation** of 246234.82 **units**
The SumAssured range is between **168536 units (Minimum)** and **1838496 units (Maximum)**
The **median** of the SumAssured is **578976.5 units**
Total count is **4366.**

LastMonthCalls

Here LastMonthCalls is actually *Total calls attempted by company to a customer for cross sell*
Average LastMonthCalls is **4.62 units** with **standard deviation** of **3.62 units**
The LastMonthCalls range is between **0.0 units (Minimum)** and **18 units (Maximum)**
The **median** of the LastMonthCalls is **3.0 units**
Total count is **4520.**

The Summary Of The Categorical Variables

```
Channel
Agent                    71.0
Third Party Partner      19.0
Online                   10.0
Name: Channel, dtype: float64
```

Channel

Here Channel **is actually** through *which acquisition of customer is done*
There are **3 unique** categories.
Agent – 71%
Third Party – 19%
Online – 10%

```
Occupation
Salaried          48.0
Small Business    42.0
Large Business     9.0
Freelancer         0.0
Name: Occupation, dtype: float64
```

Occupation

Here Occupation **is actually** *Occupation of customer*
There are **4 unique** categories.
Salaried – 48%
Small Business – 42%
Large Business – 9%
Freelancer – 0.00%

```
Gender
Male      59.0
Female    41.0
Name: Gender, dtype: float64
```

Gender

Here Gender **is actually** *Gender of customer*
There are **2 unique** categories.
Male – 59%
Female – 41%

```
Designation
Executive                    37.0
Manager                      36.0
Senior Manager               15.0
Assistant Vice President      7.0
Vice President                5.0
Name: Designation, dtype: float64
```

Designation

Here Designation **is actually *Designation*** *of customer in their organization*
There are **5 unique** categories.
Executive – 37%
Manager – 36%
Senior Manger – 15%
Assistant Vice President – 7%
Vice President – 5%

```
MaritalStatus
Married      50.0
Single       28.0
Divorced     18.0
Unmarried     4.0
Name: MaritalStatus, dtype: float64
```

Marital Status

Here Marital Status **is actually *Marital status of customer***

There are **4 unique** categories.

Married – 50%
Single – 28%
Divorced – 18%
Unmarried – 4%

```
EducationField
Graduate          41.0
Under Graduate    31.0
Diploma           11.0
Engineer           9.0
Post Graduate      6.0
MBA                2.0
Name: EducationField, dtype: float64
```

Educational Field

Here Educational Field **is actually *Field of education of customer***

There are **6 unique** categories.

Graduate – 41%
Under Graduate – 31%
Diploma – 11%
Engineer – 9%
Post Graduate – 6%
MBA – 2%

```
Zone
West      57.0
North     42.0
East       1.0
South      0.0
Name: Zone, dtype: float64
```

Zone

Here Zone **is actually *Customer belongs to which zone in India***

There are **4 unique** categories.

West – 57%
North – 42%
East – 1%
South – 0%

```
PaymentMethod
Half Yearly    59.0
Yearly         32.0
Monthly         8.0
Quarterly       2.0
Name: PaymentMethod, dtype: float64
```

Payment Method

Here Zone **is actually *Frequency of payment selected by customer***

There are **4 unique** categories.

Half Yearly – 59%
Yearly – 32%
Monthly – 8%
Quarterly – 2%

```
ExistingProdType
4    42.0
3    30.0
5    16.0
2     5.0
1     4.0
6     3.0
Name: ExistingProdType, dtype: float64
```

Existing Prod Type

Here Existing Prod Type **is actually *Existing product type of customer***

There are **6 unique** categories.

4 – 42%
3 – 30%
5 – 16%
2 – 5%
1 – 4%
6 – 3%

```
NumberOfPolicy
4.0    24.0
3.0    21.0
5.0    19.0
2.0    16.0
1.0    10.0
6.0    10.0
Name: NumberOfPolicy, dtype: float64
```

NumberOfPolicy

Here NumberOfPolicy **is actually *Total number of existing policies of a customer***

There are **6 unique** categories.

4.0– 24%
3.0 – 21%
5.0 – 19%
2.0 – 16%
1.0 – 10%
6.0 – 10%

```
Complaint
0    71.0
1    29.0
Name: Complaint, dtype: float64
```

Complaint

Here Complaint **is actually *Indicator of complaint registered in last one month by customer***

There are **2 unique** categories.

0 (No Complaint) – 71%
1(Complaints) – 29%

```
CustCareScore
3.0    31.0
1.0    21.0
5.0    20.0
4.0    18.0
2.0    10.0
Name: CustCareScore, dtype: float64
```

CustCareScore

Here CustCareScore **is actually *Customer satisfaction score given by customer in previous service call***

There are **5 unique** categories.

3.0 – 31%
1.0 – 21%
5.0 – 20%
4.0 – 18%
2.0 – 10%

c) Understanding of attributes (variable info, renaming if required)

The variable info

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4520 entries, 0 to 4519
Data columns (total 20 columns):
 #   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   object
 9   Designation              4520 non-null   object
 10  NumberOfPolicy           4475 non-null   object
 11  MaritalStatus            4520 non-null   object
 12  MonthlyIncome            4284 non-null   float64
 13  Complaint                4520 non-null   object
 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   object
dtypes: float64(5), int64(3), object(12)
memory usage: 706.4+ KB
```

Info- Details

- Most data types are object oriented followed by Float and Integers
- There is presence of missing values within the dataset

The Datatypes

```
object      12
float64      5
int64        3
dtype: int64
```

Datatypes

- The object-oriented data types are majority in nos., followed by Float64 and Int64

Renaming Process

Many of the object-oriented labels in several columns are being renamed for better understanding-

- ***Occupation***
  In this variable, the two label that are being renamed are-
  Laarge Business → Large Business
  Free Lancer → Freelancer

- ***EducationField***
  In this variable, the two label that are being renamed are-
  UG → Under Graduate

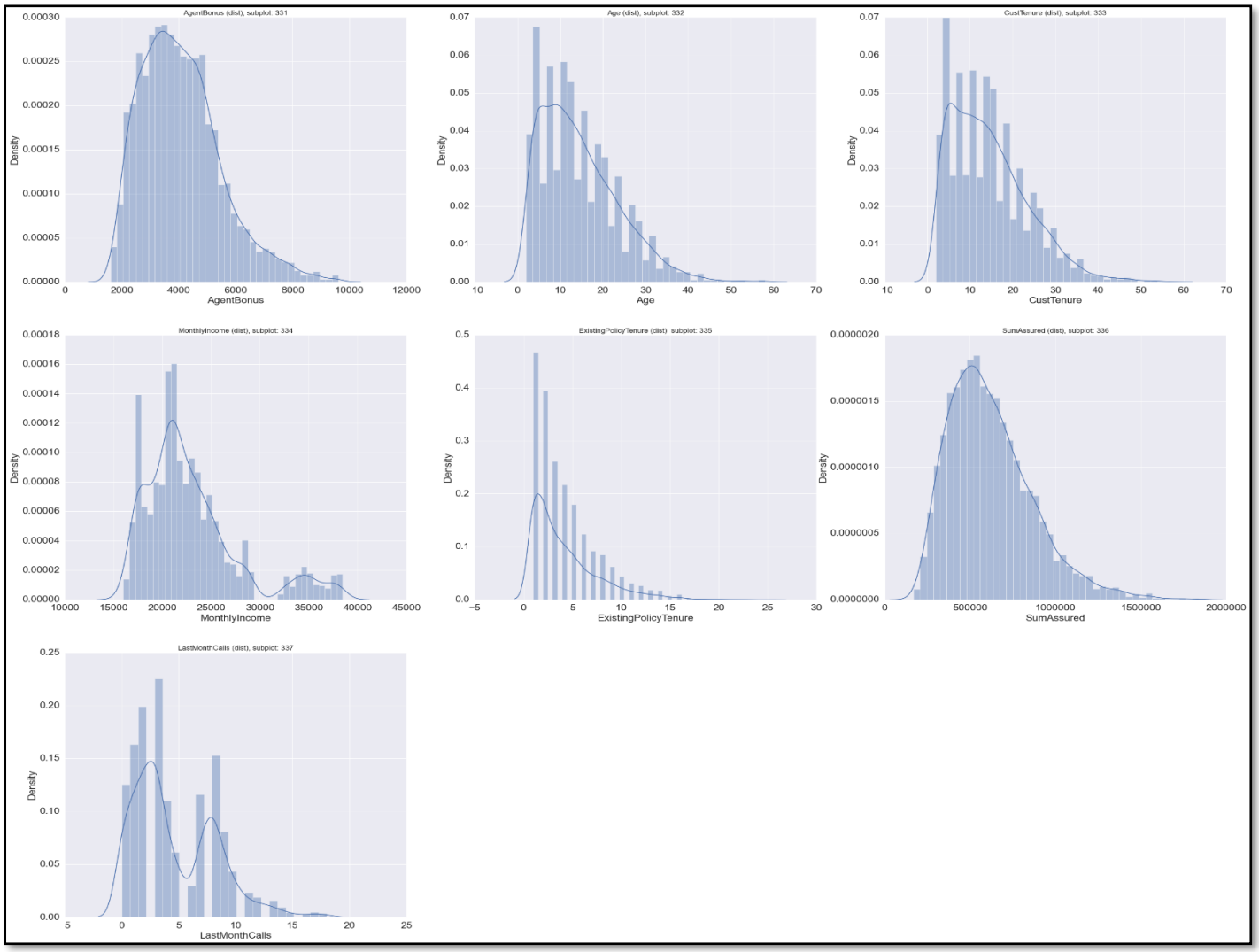
- ***Gender***
  In this variable, the two label that are being renamed are-
  Fe Male → Female

- ***Designation***
  In this variable, the two label that are being renamed are-
  Exe→ Executive
  AVP → Assistant Vice President
  VP → Vice President

Univariate analysis (distribution and spread for every continuous attribute, distribution of data in categories for categorical ones)

Univariate Analysis- The term **univariate analysis** refers to the analysis of one variable. You can remember this because the prefix "uni" means "one."
The purpose of univariate analysis is to understand the distribution of values for a single variable.

Distribution plots (histogram)



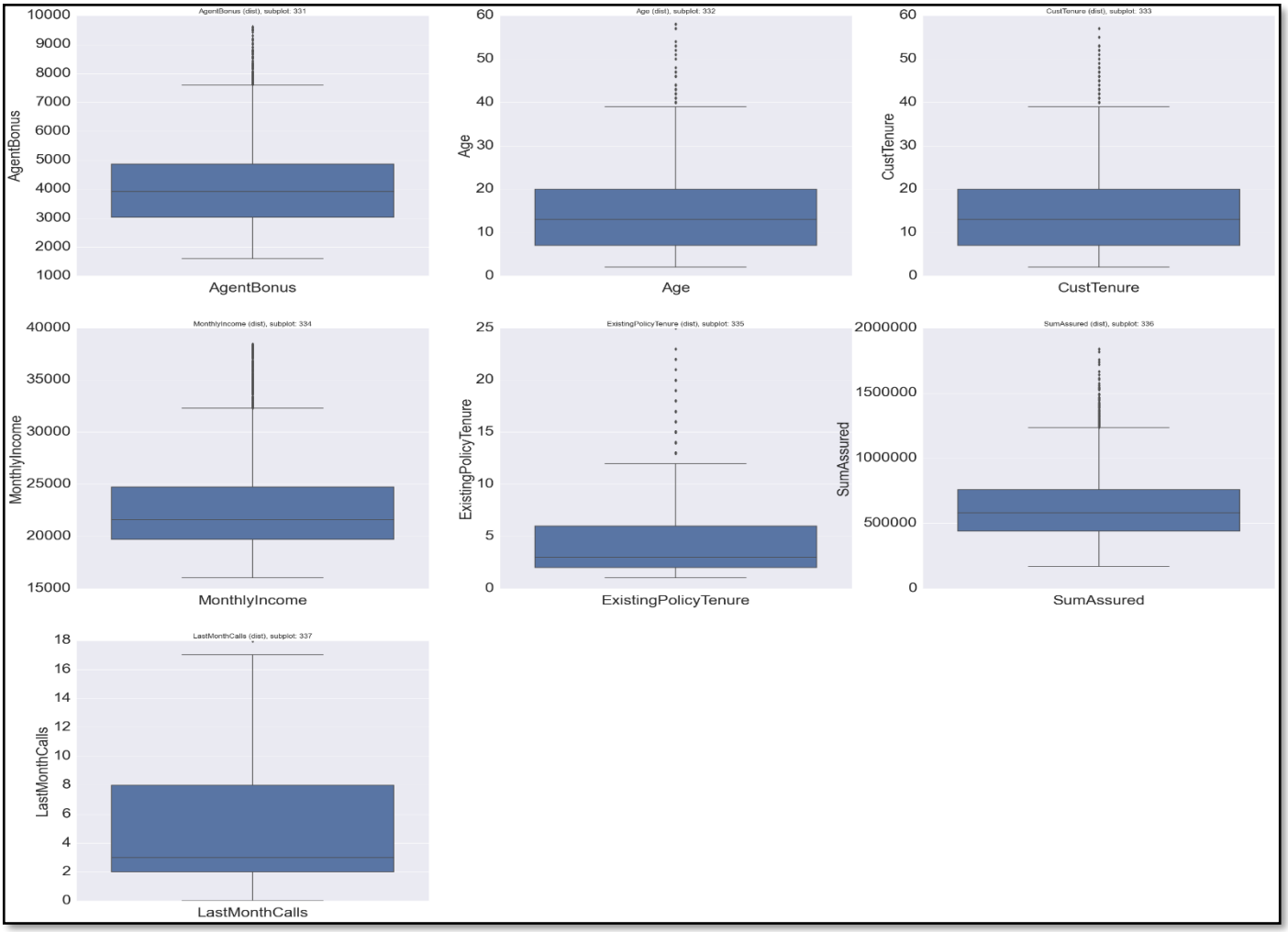
Conclusion- From the above histogram plot of the continuous variables, we can say that-

- Most of the numeric variables are having positive skewness or are right skewed
- Both Existing policy and last month calls are having discrete variable types, hence we can observe gaps in between certain frequencies
- AgentBonus- The distribution here seems to have positive skewness. The maximum distribution is between the ranges 2000-6000 units.
- Age- The distribution here seems to have positive skewness. There seems to be fluctuations in the distribution of Age overall. The maximum distribution is between the ranges 5-20 units.

- CustTenure- The distribution here seems to have positive skewness. There seems to be fluctuations in the distribution of CustTenure overall. The maximum distribution is between the ranges 5-20 units.
- MonthlyIncome- The distribution here seems to have positive skewness with bimodal appearance. The two ranges are –(15000-30000) units and (31000-38000)units approx
- ExistingPolicyTenure- The distribution here seems to have positive skewness. There seems to be fluctuations in the distribution of ExistingPolicyTenure overall. The maximum distribution is between the ranges 1-3 units.
- SumAssured- Age- The distribution here seems to have positive skewness. The maximum distribution is between the ranges 100000-1000000 units.
- LastMonthCalls- Here we can observe that the distribution has been in blocks, with gaps in between, causing a bimodal appearance. The two ranges are –(0-5) units and (6-10)units approx

---

Box plots (without outlier treatment) for the continuous columns



Conclusion- - From the above box plot of the continuous variables, we can say that-

- Most of the numeric variables are having outliers except for LastMonthCalls
- AgentBonus- The boxplot shows a large no. of outliers in this category, with positive skewness, with median lies at 4000 units
- Age- The boxplot shows a large no. of outliers in this category, with highly positive skewness, with median lies at approx. 12 units
- CustTenure- The boxplot shows a large no. of outliers in this category, with highly positive skewness, with median lies at approx. 12 units
- MonthlyIncome- The boxplot shows a large no. of outliers in this category, with highly positive skewness, with median lies at approx. 22000 units
- ExistingPolicyTenure- The boxplot shows a large no. of outliers in this category, with highly positive skewness, with median lies at approx. 4 units
- SumAssured-The boxplot shows a large no. of outliers in this category, with positive skewness, with median lies at approx. 600000 units
- LastMonthCalls- The boxplot shows no outliers in this category, with positive skewness, with median lies at approx. 3 units

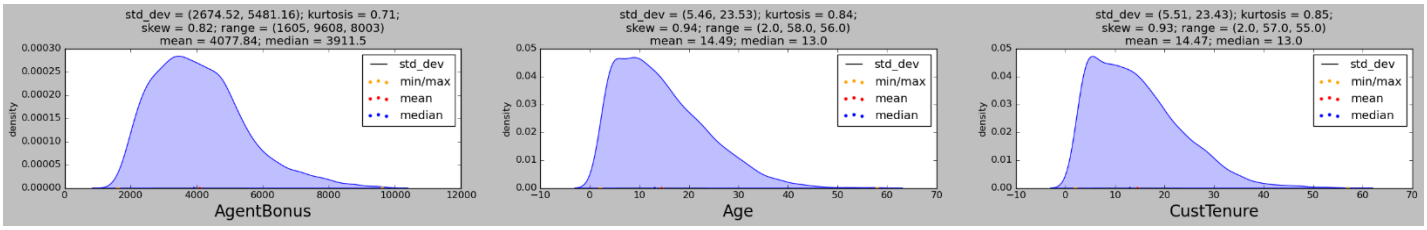
---

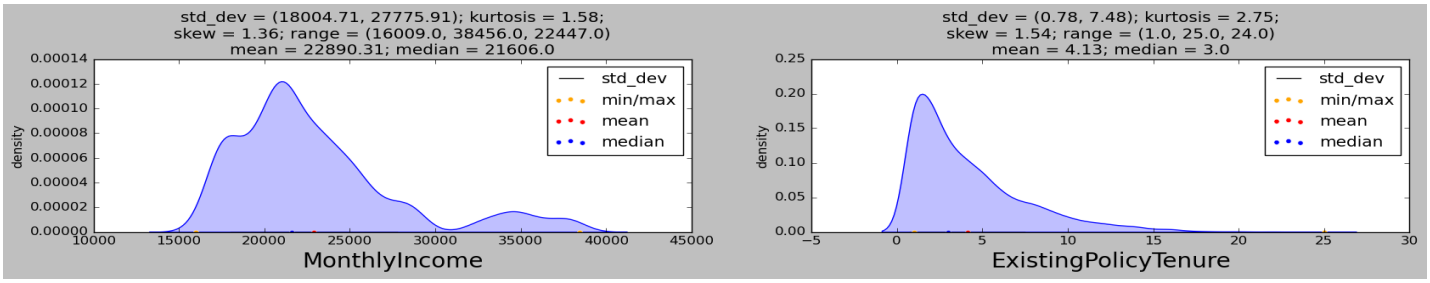
Skewness -Skewness is a measure of symmetry, or more precisely, the lack of symmetry. A distribution, or data set, is symmetric if it looks the same to the left and right of the centre point.

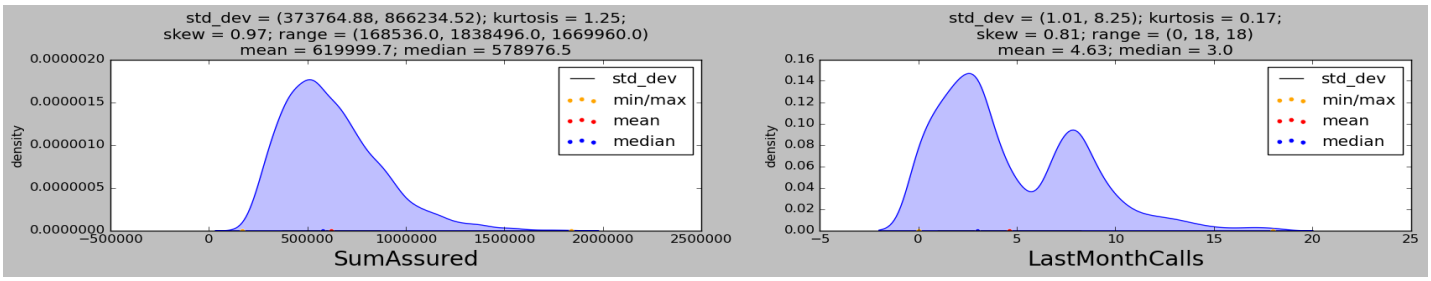
If the skewness is between -0.5 and 0.5, the data are fairly symmetrical. If the skewness is between -1 and – 0.5 or between 0.5 and 1, the data are moderately skewed. If the skewness is less than -1 or greater than 1, the data are highly skewed.

Kurtosis - Kurtosis is a measure of whether the data are heavy-tailed or light-tailed relative to a normal distribution.

Skewness of the Continuous variables



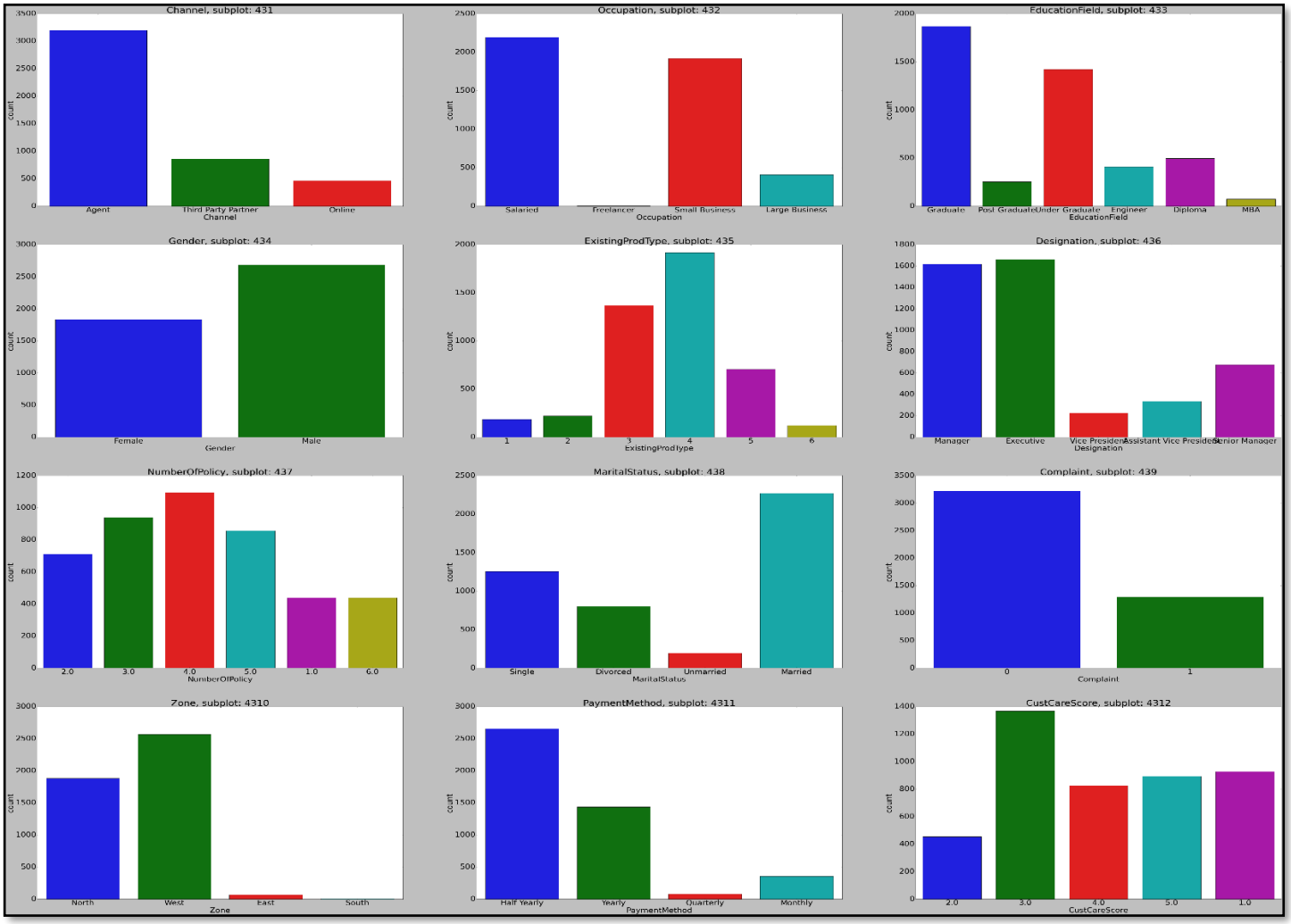




Conclusion- From the above skewness plot of the continuous variables, we can conclude that-

- Most of the numeric variables are having skewness < 1 except for MonthlyIncome and ExistingPolicyTenure
- Moderately Skewed – The variables such as AgentBonus, Age, CustTenure, SumAssured, LastMonthCalls, have values between 0.50 and 1 units, which indicate moderate skewness.
- Highly Skewed – The variables such as MonthlyIncome, ExistingPolicyTenure have values greater than 1 units, which indicate high skewness.
-

Bar plots for the categorical columns



Conclusion **-** From the above barplots of the categorical variables, we can conclude that-

- In the first bar graph, we can observe different categories of 'Channel' attribute. Most frequent is the Agent category, followed by Third Party Partner, with least frequency in the Online Category
- In the second bar graph, we can observe different categories of 'Occupation' attribute. Most frequent is the Salaried category, followed by Small Business, with least frequency in the Freelancer Category
- In the third bar graph, we can observe different categories of 'Education Field' attribute. Most frequent is the Graduate category, followed by Undergraduate, with least frequency in the MBA Category
- In the fourth bar graph, we can observe different categories of 'Gender' attribute. Most frequent is the Male category, with least frequency in the Female Category
- In the fifth bar graph, we can observe different categories of 'ExistingProdType' attribute. Most frequent is the '4th' category, followed by '3rd', with least frequency being the '6th' Category

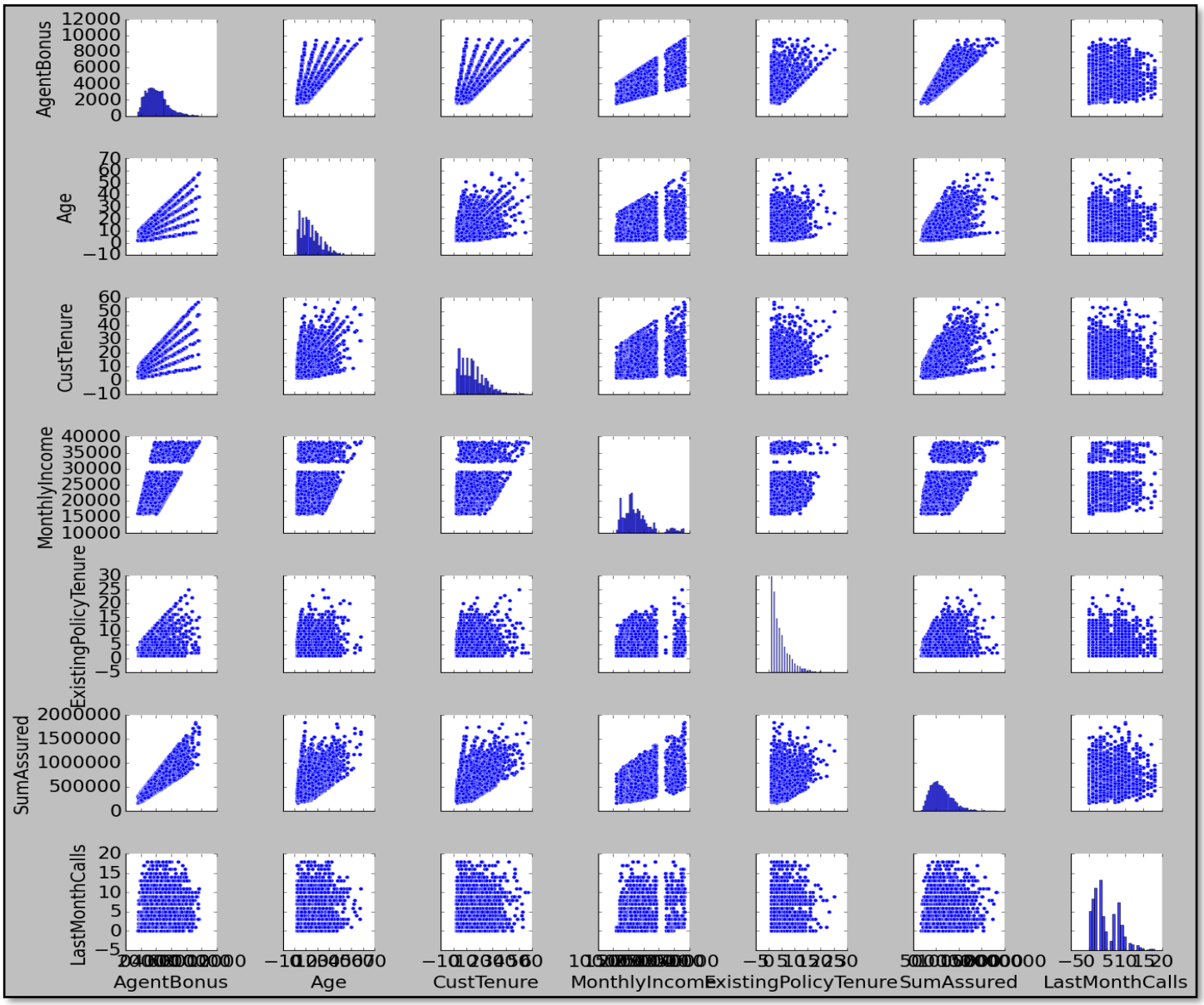
- In the sixth bar graph, we can observe different categories of 'Designation' attribute. Most frequent is the 'Executive' category, followed by 'Manager', with least frequency in the 'Vice President' category
- In the seventh bar graph, we can observe different categories of 'NumberofPolicies' attribute. Most frequently, people opt for 4 policies on an average, followed by 3 policies, with least frequency in a single policy number
- In the eighth bar graph, we can observe different categories of 'MaritalStatus 'attribute. Most frequent is the 'Married' category, followed by those who are 'Single', with least frequency observed in 'Unmarried' category
- In the ninth bar graph, we can observe different categories of 'Complaint' attribute. Most frequently, people don't complaint in general for most of the policies, which is evident from here
- In the tenth bar graph, we can observe different categories of 'Zone' attribute. Most frequently, the customers are from 'West' zone, followed by 'North', with least frequency associated with 'South'
- In the eleventh bar graph, we can observe different categories of 'PaymentMethod' attribute. Most frequently, the customers opt for 'HalfYearly' payment method, followed by 'Yearly', with least chosen method being 'Quarterly'
- In the twelfth bar graph, we can observe different categories of 'CustCareScore' attribute. Most frequently, the customers provide a score of 3, followed by a score of 1, with least chosen score being 2

---

a) Bivariate analysis (relationship between different variables , correlations)

The term **bivariate analysis** involves the analysis of two variables, for the purpose of determining the empirical relationship between them. Bivariate analysis can be helpful in testing simple hypotheses of association.

Pair plots for Bi variate data



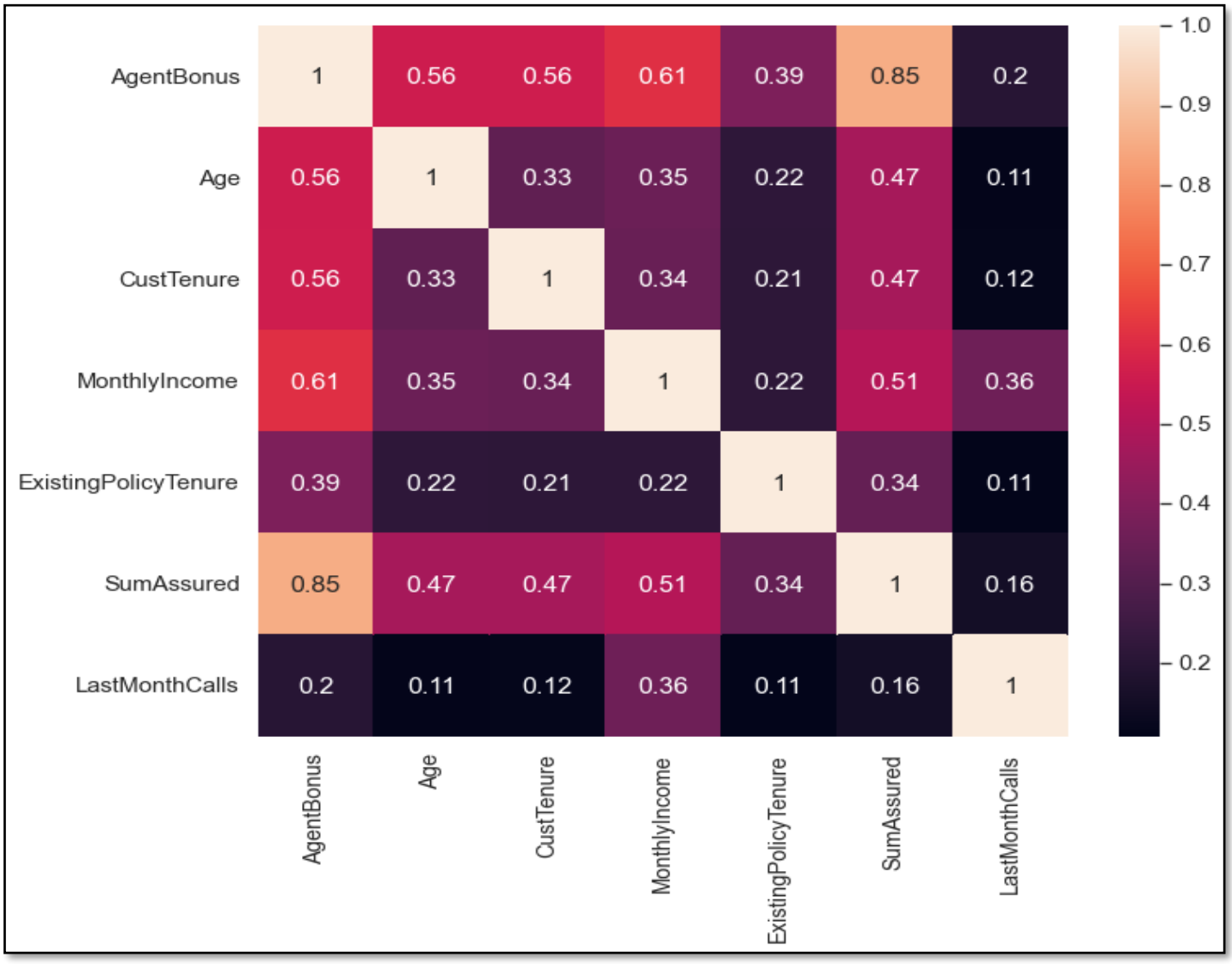
Conclusion - From the above scatterplots of the continuous variables, we can conclude that-

- AgentBonus has increased more or less with increase in the Age, CustTenure, MonthlyIncome along with SumAssured, while against the LastMonthCall, it has been non-significant or neutral
-

HeatMap

Heatmap or Corrleation plot is basically being used to evaluate the **relationship** between the **different numeric variables within a dataset**

Heat map with only continuous variables



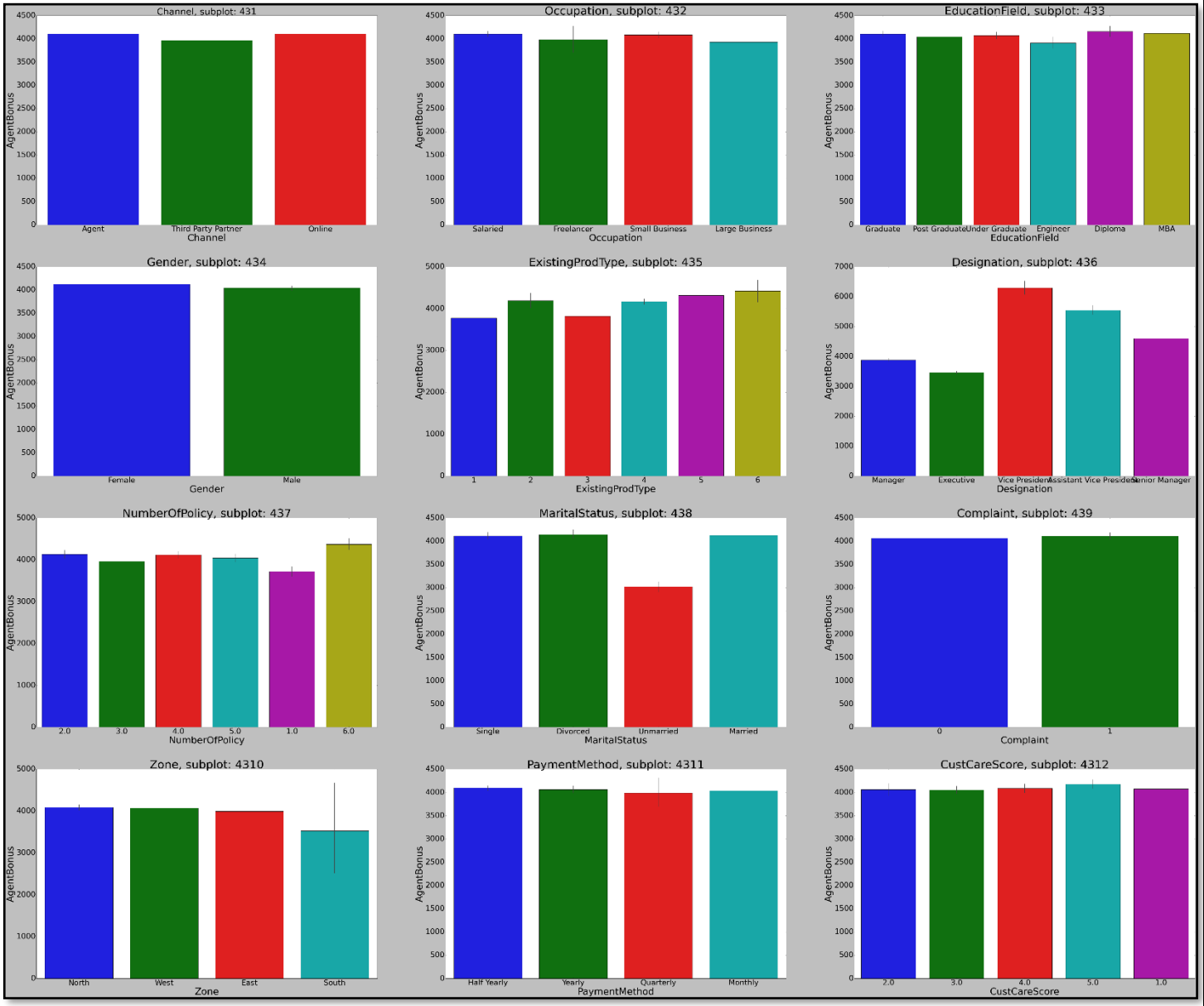
Conclusion **-** From the above heatmap of the continuous variables, we can conclude that-

- Presence of correlation can be observed for AgentBonus with that of the attributes like- SumAssured ,MonthlyIncome ,age ,CustTenure while that with ExistingPolicyTenure and LastMonthCalls it is low.
- In case of multicollinearity, it can be observed in many of the independent variables.
- Age has moderate correlation with that of the independent features like SumAssured, CustTenure, MonthlyIncome.
- CustTenure has moderate correlation with that of the independent features like SumAssured, Age, MonthlyIncome.
- MonthlyIncome has moderate correlation with that of the independent features like SumAssured, LastMonthCalls, CustTenure, Age

- ExistingPolicyTenure has moderate correlation with that of the independent features like SumAssured
- SumAssured has moderate correlation with that of the independent features like MonthlyIncome, ExistingPolicyTenure, CustTenure, Age
- LastMonthCalls has moderate correlation with that of the independent features like MonthlyIncome

---

Bar plots for the categorical columns w.r.t. AgentBonus



Conclusion - From the above Bar of the categorical variables, we can conclude that-

- In the first bar graph, we can observe different categories of 'Channel' attribute against the AgentBonus variable. Most contribution to AgentBonus is done by Agent Channel type as well as Online channel with least contribution from Third Party Channel
- In the second bar graph, we can observe different categories of 'Occupation' attribute against the AgentBonus variable. Most contribution to AgentBonus is done by Small Business type as well as Salaried with least contribution from Large Business
- In the third bar graph, we can observe different categories of 'EducationField' attribute against the AgentBonus variable. Major contributions to AgentBonus are by Diploma Holders as well as Graduate ones with least contribution from Large Business

- In the fourth bar graph, we can observe different categories of 'Gender' attribute against the AgentBonus variable. Female agents have received more Agent Bonus than the Male counterparts.
- In the fifth bar graph, we can observe different categories of 'ExistingProdType' attribute against the AgentBonus variable. In terms of the Product categories, Category 6th has contributed in majority to the Agent Bonus amount followed by 5th category, while least contribution was by 3rd and 1st categories
- In the sixth bar graph, we can observe different categories of 'Designation' attribute against the AgentBonus variable. In terms of the Designation categories, for the VicePresident category,most Agent Bonus amount is received followed by Assistant Vice President, while least bonus is being received against Executive
- In the seventh bar graph, we can observe different categories of 'No. of Policy' attribute against the AgentBonus variable. In terms of the No of policies, Highest bonus is being linked with Highest no. of policies -6, while least amount of Bonus is linked to least no. of policies- 1
- In the eighth bar graph, we can observe different categories of 'MaritalStatus' attribute against the AgentBonus variable. Those who are Divorced or Married, are responsible for majority of the Agent Bonus, while the least contribution in this part is by Unmarried customers.
- In the ninth bar graph, we can observe different categories of 'Complaint' attribute against the AgentBonus variable. Here, Complaint types don't have any significance on the Agent Bonus that much.
- In the tenth bar graph, we can observe different categories of 'Zone' attribute against the AgentBonus variable. The Customers from the North and West Segment have contributed to more Bonus as compared to the South.
- In the eleventh bar graph, we can observe different categories of 'PaymentMethod' attribute against the AgentBonus variable. Here, more or less all the different Payment Methods have contributed to the receival of Agent Bonus
- In the twelveth bar graph, we can observe different categories of 'CustCareScore' attribute against the AgentBonus variable. Here, the highest bonus is being linked with Highest Customer Care Score -5, while least amount of Bonus is linked to the rest.

****

3. Data Cleaning and Pre-processing

- Approach used for identifying and treating missing values and outlier treatment (and why)
- Need for variable transformation (if any)
- Variables removed or added and why (if any)

a) Missing Value treatment (if applicable)

The missing values or 'NaN' or 'NA' values are in general to be cleaned during the data cleaning process.

They are basically –

1. Removed
2. Replaced with mode function (Categorical variables)
3. Replaced with Median function (Continuous variables)
4. Replace with other processes

In this dataset, the no of missing values in different variables are shown below -

Missing Values

Age	269
MonthlyIncome	236
CustTenure	226
ExistingPolicyTenure	184
SumAssured	154
CustCareScore	52
NumberOfPolicy	45
dtype: int64	

Categorical Variables

We will r*eplace the categorical variables with the mode function*
The variables are-

- CustCareScore
- NumberOfPolicy

Continuous Variables

We will r*eplace the continuous variables with the median function*
The variables are-

- Age
- MonthlyIncome
- CustTenure
- ExistingPolicyTenure
- SumAssured

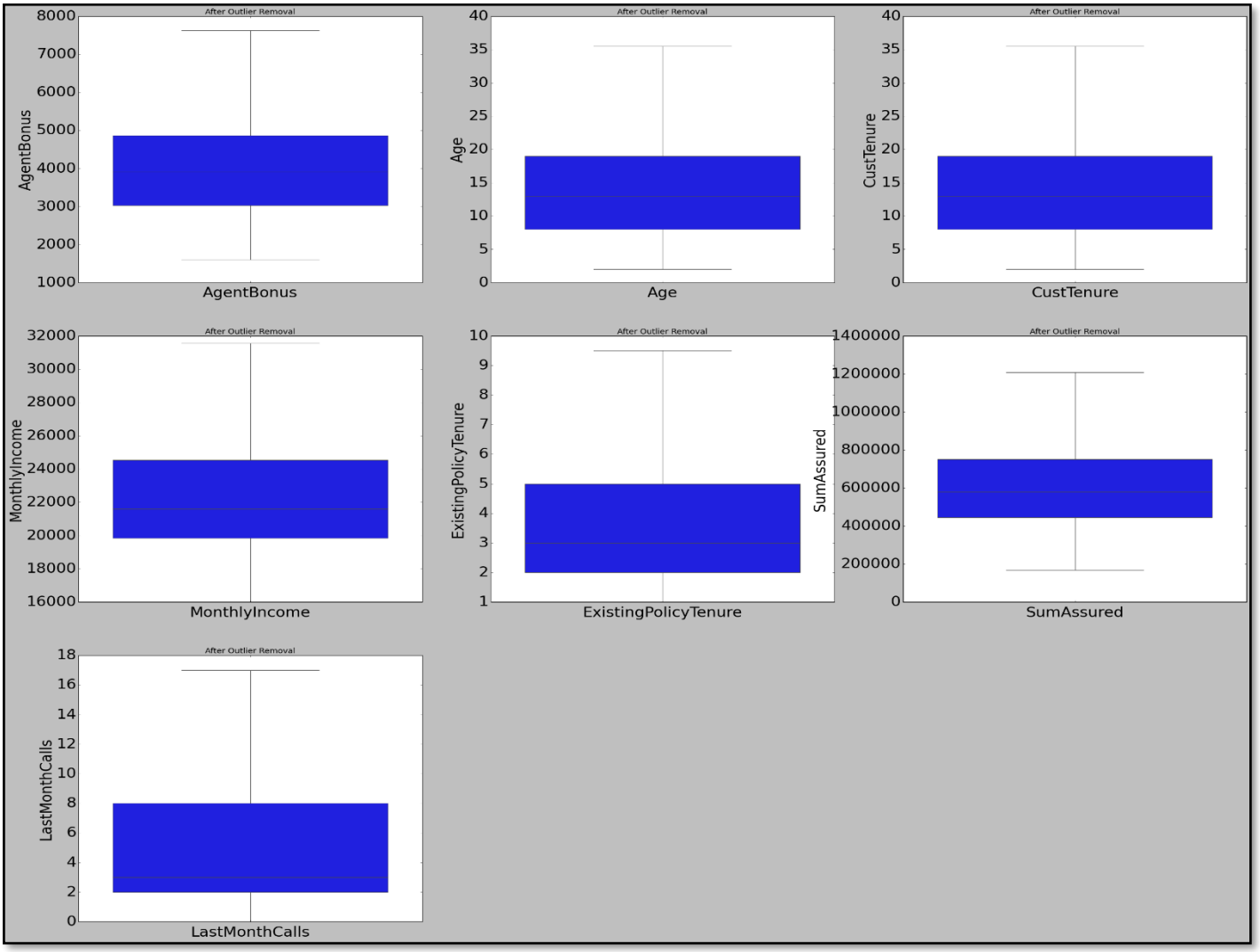
---

b) Outlier treatment (if required)

Outliers are basically the extreme values in the dataset. Outliers increase the variability in your data, which decreases statistical power.

As the dataset has huge no. of outliers in most of the continuous variables, we created a user defined function to remove the outliers from the data. The boxplot after outlier removal is show below.

Box plots (with outlier treatment) for the continuous columns



Conclusion- It can be checked from the above graph that the outliers for each of the numeric attributes are being treated using a user defined function

---

c) Variable transformation (if applicable)

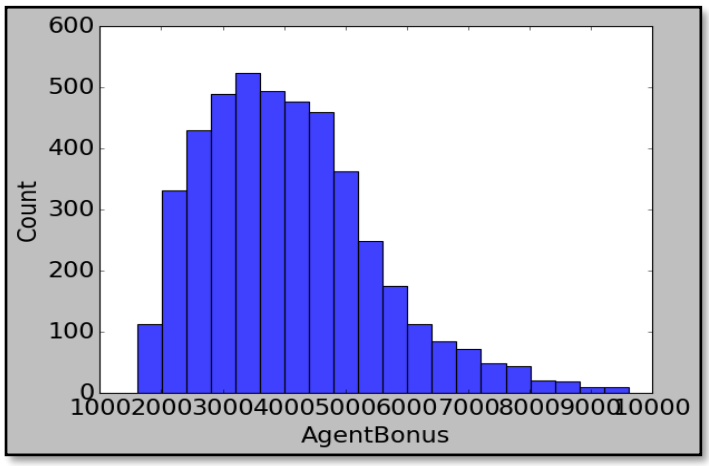
Yes, we need to transform a certain variable which is the target variable here- AgentBonus.
Method of treatment- We will be using Log Transformation

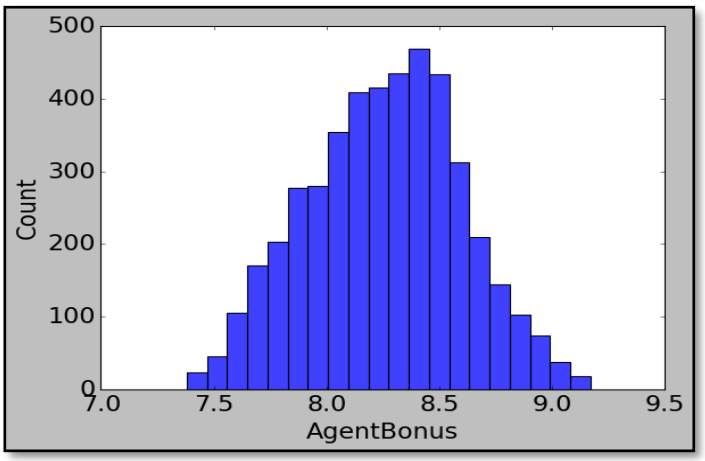
Log-Transformation

Log transformation is a data transformation method in which it replaces each variable x with a log(x). The choice of the logarithm base is usually left up to the analyst and it would depend on the purposes of statistical modelling. In this article, we will focus on the natural log transformation. The nature log is denoted as ln.
When our original continuous data do not follow the bell curve, we can log transform this data to make it as "normal" as possible so that the statistical analysis results from this data become more valid. In other words, the log transformation reduces or removes the skewness of our original data.

Log transformation of the AgentBonus variable looks to be slightly more symmetrically distributed. We can use a log of the AgentBonus variable as our target variable in the regression model, to check if performance is better than the AgentBonus feature used without any transformation

Variable Transformation - AgentBonus





Before transformation

After transformation

---

d) Addition or Removal of unwanted variables (if applicable)

We have dropped the following variable before moving into EDA as they provide limited significance to the problem,

- ***CustID***

There has been no Addition of New Variables

****

4. Model building

- Clear on why was a particular model(s) chosen.

- Effort to improve model performance.

Encoding

Feature engineering is the most important aspect of a data science model development. There are several categories of features in a raw dataset. Features can be text, date/time, categorical, and continuous variables. For a machine learning model, the dataset needs to be processed in the form of numerical vectors to train it using an ML algorithm.

The feature engineering techniques are being used to transform a categorical feature into a continuous feature and vice-versa.

The processes being involved are-

- **Feature Binning:** Conversion of a continuous variable to categorical.
- **Feature Encoding:** Conversion of a categorical variable to numerical features.

In this project we will discuss about Feature Encoding process, after describing the type of categorical variables present in the dataset.

Most of the Machine learning algorithms cannot handle categorical variables unless we convert them to numerical values. Many algorithm's performances vary based on how Categorical variables are encoded.

Categorical variables can be divided into three categories: Binary (only two types), Nominal (No particular order) and Ordinal (some ordered).

- Binary – A Binary variable has only values or categories. For example, Yes/No, True/False
- Nominal - A nominal variable has no intrinsic ordering to its categories. For example, gender is a categorical variable having two categories (male and female) with no intrinsic ordering to the categories.
- Ordinal - An ordinal variable has a clear ordering. For example, temperature as a variable with three *orderly* categories (low, medium and high).

In this case-

- Binary categories - Gender
- Ordinal categories – Designation, Educational Field, Customer Care Score
- Nominal Categories - Channel, Occupation, ExistingProdType, Designation, NumberOfPolicy, MaritalStatus, Complaint, Zone, PaymentMethod

PS. Customer Care Score, ExistingProdType, NumberOfPolicy, Complaint – all these are already discrete variables which are considered categorical in nature from the very start

We first converted them from Numerical type (int64) to Object type(object) before carrying out with the EDA, as they were not actually numerical in literal terms but are basically the ordinal data type.

Before splitting the data for model creation, we have further converted those object datatypes into int64 format for the ease in the process as required.

There are many ways we can encode the remaining categorical variables as numbers and use them in an algorithm. We will describe the processes as follows-

Label Encoding

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.

Example:

Suppose we have a column *Height* in some dataset.

Height
Tall
Medium
Short

Before applying label encoding

Height
0
1
2

After applying label encoding

where 0 is the label for tall, 1 is the label for medium, and 2 is a label for short height.

Pros

- It is easy to implement and interpret
- It is visually user friendly
- It works best with smaller number of unique categorical values

Cons

- Depending upon the data values and type of data, label encoding induces a new problem since it uses number sequencing. The problem using the number is that they introduce relation/comparison between them.
- It 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.
- It can skew the estimation results if there is a large number of unique categorical values. In our case it was 4, but if it's 10 or more, you should keep this in mind.

To use **Label encoder** on categorical variables**,** we have imported **LabelEncoder module** from **sklearn.preprocessing package**

One Hot encoding

To overcome the Disadvantage of Label Encoding as it considers some hierarchy in the columns which can be misleading to nominal features present in the data. we can use the One-Hot Encoding strategy.

In this technique, the categorical parameters will prepare separate columns for individual labels. So, wherever there is A category, the value will be 1 in A column and 0 in B column, and vice-versa. Let's understand with an example

One-hot encoding is processed in 2 steps:

1. Splitting of categories into different columns.
2. Put '0 for others and '1' as an indicator for the appropriate column.

Pros

- One-Hot-Encoding has the advantage that the result is binary rather than ordinal and that everything sits in an orthogonal vector space.

Cons

- The disadvantage is that for high cardinality, the feature space can really blow up quickly and you start fighting with the curse of dimensionality.

We have used **pd.get_dummies** to convert them into **numerical types (int8)**

When considering **One Hot Encoding** (OHE) and **Label Encoding**, we must try and understand what model you are trying to build. Namely the two categories of model we will be considering are:

1. **Tree Based Models**: Gradient Boosted, Decision Trees and Random Forests.
2. **Non-Tree Based Models**: Linear, KNN or Neural Network based.

Let's consider when to apply OHE and when to apply Label Encoding while building tree-based models.

We apply OHE when:

- When the values that are **close to each other** in the label encoding correspond to target values that aren't close (non-linear data).
- When the categorical **feature is not ordinal** (dog, cat, mouse).

We apply Label encoding when-

1. The categorical **feature is ordinal** (Jr. kg, Sr. kg, Primary school, high school, etc).
2. When we can come up with a label encoder that **assigns close labels to similar categories**: This leads to less splits in the trees hence reducing the execution time.
3. When the number of categorical features in the dataset is huge: One-hot encoding a categorical feature with huge number of values can lead to (1) high memory consumption and (2) the case when non-categorical features are rarely used by model. You can deal with

the 1st case if you employ sparse matrices. The 2nd case can occur if you build a tree using only a subset of features. For example, if you have 9 numeric features and 1 categorical with 100 unique values and you one-hot-encoded that categorical feature, you will get 109 features. If a tree is built with only a subset of features, initial 9 numeric features will rarely be used. In this case, you can increase the parameter controlling size of this subset. In xgboost it is called colsample_bytree, in sklearn's Random Forest max_features.

We have used **Manual or user defined encoding process** for two Categorical Variables because of their ordinal nature-

- Designation
- EducationalField

Designation	
Vice President	1
Assistant Vice President	2
Senior Manager	3
Manager	4
Executive	5

Educational Field	
MBA	1
Post Graduate	2
Engineer	3
Graduate	4
Diploma	5
Under Graduate	6

Scaling

Feature scaling in machine learning is one of the most critical steps during the pre-processing of data before creating a machine learning model. Scaling can make a difference between a weak machine learning model and a better one.
Feature Scaling is the process of changing the scale of certain features to a common one.

Machine learning algorithm just sees number — if there is a vast difference in the range say few ranging in thousands and few ranging in the tens, and it makes the underlying assumption that higher ranging numbers have superiority of some sort. So, these more significant number starts playing a more decisive role while training the model.

- **Gradient Descent Based Algorithms**

  Machine learning algorithms like **linear regression, logistic regression, neural network, etc.** that use gradient descent as an optimization technique **require data to be scaled.**
- **Distance-Based Algorithms**

  Distance algorithms like **KNN, K-means, and SVM** are **most affected** by the range of features. This is because behind the scenes **they are using distances between data points to determine their similarity.**
- **Tree-Based Algorithms**

  Tree-based algorithms, on the other hand, are fairly **insensitive** to the scale of the features.

- Algorithms like **Linear Discriminant Analysis (LDA), Naive Bayes is** by design equipped to handle this and give weights to the features accordingly. Performing features scaling in these algorithms may not have much effect.
- Scaling is critical while performing **Principal Component Analysis (PCA)**. PCA tries to get the features with maximum variance, and the variance is high for high magnitude features and skews the PCA towards high magnitude features.

Type of Scaling Technique present -

Normalization -Normalization is the process of scaling data into a range of [0, 1]. It's more useful and common for regression tasks.

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}}$$

Standardization - Standardization is the process of scaling data so that they have a mean value of 0 and a standard deviation of 1. It's more useful and common for classification tasks.

$$x' = \frac{x - \mu}{\sigma}$$

Type of Scaling Technique used –

For Linear and KNN regressor, Min Max scalar will be used

We will scale the data based on Min Max Scalar or Normalization technique in SkLearn

- Normalization is also known as Min-Max Scaling and Scikit-Learn provides the MinMaxScaler for this purpose. On the other hand, it also provides a Normalizer, which can make things a bit confusing.

Note: The Normalizer class doesn't perform the same scaling MinMaxScaler.
Normalizer works on rows, not features, and it scales them independently.

- Here, the minimum of feature is made equal to zero and the maximum of feature equal to one. MinMax Scaler shrinks the data within the given range, usually of 0 to 1. It transforms data by scaling features to a given range. It scales the values to a specific value range without changing the shape of the original distribution.

For Ridge and Lasso Regression, Standard Scalar to be used

We will scale the data based on Z-Score method or Standard Scalar in SkLearn

- Standard Scalar standardizes a feature by subtracting the mean and then scaling to unit variance. Unit variance means dividing all the values by the standard deviation.
- Standard Scalar results in a distribution with a standard deviation equal to 1. The variance is equal to 1 also, because variance = standard deviation squared. And 1 squared = 1.
- Standard Scalar makes the mean of the distribution 0. About 68% of the values will lie be between -1 and 1.

Standard Scalar normalizes the data using the formula (x-mean)/standard deviation.

Z = (value -mean)/standard deviation

Why it is to be used?

MinMax Scalar- MinMax Scaler shrinks the data within the given range, usually of 0 to 1. It transforms data by scaling features to a given range. It scales the values to a specific value range without changing the shape of the original distribution.

Standard Scalar- If your variables are of incomparable units (e.g. height in cm and weight in kg) then you should standardize variables, of course. Even if variables are of the same units but show quite different variances it is still a good idea to standardize **Gradient Descent Based Algorithms** and **Distance-Based Algorithms.**

Why to normalize prior to model fitting?

The main idea behind normalization/standardization is always the same. **Variables** that are **measured** at **different scales do not contribute equally to the model fitting** & model learned function and might end up creating a **bias**. Thus, to deal with this potential problem feature-wise normalization such as **MinMax** Scaling is usually used prior to model fitting.

Explanation with respect to the Variables given

As per the data provided, standard scaling is required as all the different variables are provided in different units, for example, Age is a continuous variable in years/months, similar to that of Cust_Tenure, ExistingPolicyTenure as all of these are associate with Date. While MonthlyIncome, SumAssured are continuous in nature and they define the Amount of

Money. As, there are differences in units, hence other values expressed in higher units will outweigh the variables in lower units and can give varied results. This is why scaling is important, and Standard scalar, as mentioned above, normalise the data points with mean 0 and standard deviation 1.

Pros of Min Max Scalar

- **Normalization** is about **transforming** the feature values to fall within the **bounded intervals (min and max)**

Cons of Min Max Scalar

- Min-max normalization has one fairly significant downside: **it does not handle outliers very well**. For example, if you have 99 values between 0 and 40, and one value is 100, then the 99 values will all be transformed to a value between 0 and 0.4. That data is just as squished as before!

Pros of Standard Scalar

- Unlike Normalization, standardization maintains useful information about outliers and makes the algorithm less sensitive to them

---

Metrices to be used

❖ ***R - Squared***

- R-squared (R^2) is a statistical measure that **represents the proportion of the variance for a dependent variable** that's explained by an independent variable or variables in a regression model.
- It is also called the coefficient of determination is used to evaluate the performance of a linear regression model.
- What qualifies as a "good" R-Squared value will depend on the context. In some fields, such as the social sciences, even a relatively low R-Squared such as 0.5 could be considered relatively strong. In other fields, the standards for a good R-Squared reading can be much higher, such as 0.9 or above.
- Essentially, an R-Squared value of 0.9 would indicate that 90% of the variance of the dependent variable being studied is explained by the variance of the independent variable.

❖ ***Adjusted R^2 measure***

- Adjusted R-squared is a **modified version of R-squared** that has been adjusted for the number of predictors in the model.
- The adjusted R-squared increases when the new term improves the model more than would be expected by chance. It decreases when a predictor improves the model by less than expected.
- Typically, the adjusted R-squared is positive, not negative. It is always **lower than the R-squared**.

❖ ***RMSE***

- **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 lower the value of the **Root Mean Squared Error**, the better the model is. A perfect model (a hypothetic model that would always predict the exact expected value) would have a **Root Mean Squared Error** value of 0.
- The **Root Mean Squared Error** has the advantage of representing the amount of error in the same unit as the predicted column making it easy to interpret. If you are trying to predict an amount in dollars, then the **Root Mean Squared Error** can be interpreted as the amount of error in dollars

❖ ***MSE***

- The **mean squared error** (MSE) tells us how close a regression line is to a set of points. It does this by taking the distances from the points to the regression line (these distances are the "errors") and squaring them. The squaring is necessary to remove any negative signs. It also gives more weight to larger differences.
- It's called the **mean** squared error as we're finding the average of a set of errors.
- A larger MSE indicates that the data points are dispersed widely around its central moment (mean), whereas a smaller MSE suggests the opposite.
- A smaller MSE is preferred because it indicates that your data points are dispersed closely around its central moment (mean).
- It reflects the centralized distribution of your data values, the fact that it is not skewed, and, most importantly, it has fewer errors (errors measured by the dispersion of the data points from its mean).
- Lesser the MSE => Smaller is the error => Better the estimator.

MSE formula = $(1/n) * \Sigma(\text{actual} - \text{forecast})^2$
Where:

- n = number of items,
- Σ = summation notation,
- Actual = original or observed y-value,
- Forecast = y-value from regression.

❖ ***MAPE***

- The **mean absolute percentage error** (MAPE) — also called the mean absolute percentage deviation (MAPD) — measures accuracy of a forecast system. It measures this accuracy as a percentage, and can be calculated as the average absolute percent error for each time period minus actual values divided by actual values.
- It comes under percentage errors which are scale independent and can be used for comparing series on different scales.
- 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.
- The disadvantage of MAPE is that it becomes undefined if the value for actual figure is 0 for any observation in the data.

$$MAPE = mean(|e_i|/y_i) * 100$$

where e_i is the error term and y_i is the actual data at time i.

❖ ***MAE***

- **Mean Absolute Error (MAE):** MAE measures the average magnitude of the errors in a set of predictions, without considering their direction.
- It's the average over the test sample of the absolute differences between prediction and actual observation where all individual differences have equal weight.

$$\mathrm{MAE} = \frac{1}{n}\sum_{j=1}^{n}|y_j - \hat{y}_j|$$

- MAE express average model prediction error in units of the variable of interest. It can range from 0 to ∞ and is indifferent to the direction of errors. These are negatively-oriented scores, which means lower values are better.
- **A low MAE means that the measurements from the device under test are very close in absolute value to the measurements from the reference instrument**. A high MAE means that the measurements from the device under test are very far in absolute value from the measurements from the reference instrument.

****

4a. Clear on why was a particular model(s) chosen

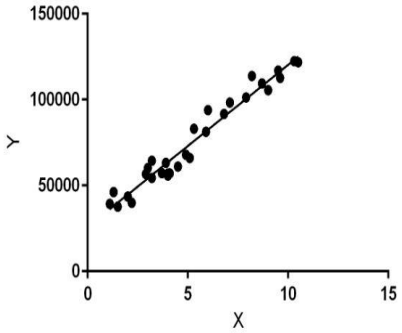
Model building and interpretation.

We will be using several ML techniques in this Supervised problem (Regression).
The different models that we will be using are

- Linear Regression
- Ridge Regression
- Lasso Regression
- Decision Tree Regressor
- Random Forest
- KNN Regression
- GBM Regressor
- XGBoosting Regressor

LINEAR REGRESSION

Linear Regression is a machine learning algorithm based on supervised learning. It performs a regression task. Regression models a target prediction value based on independent variables. It is mostly used for finding out the relationship between variables and forecasting. Different regression models differ based on – the kind of relationship between dependent and independent variables they are considering, and the number of independent variables getting used



Linear regression performs the task to predict a dependent variable value (y) based on a given independent variable (x). So, this regression technique finds out a linear relationship between x (input) and y(output). Hence, the name is Linear Regression.

In the figure above, X (input) is the work experience and Y (output) is the salary of a person. The regression line is the best fit line for our model.

Hypothesis function for Linear Regression:

$$y = \theta_1 + \theta_2.x$$

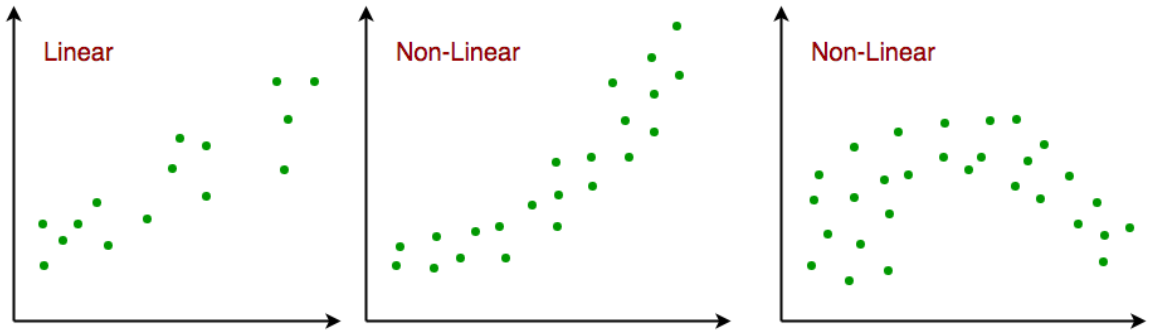
While training the model we are given:
x: input training data (univariate – one input variable(parameter))
y: labels to data (supervised learning)
When training the model – it fits the best line to predict the value of y for a given value of x. The model gets the best regression fit line by finding the best θ_1 and θ_2 values.
θ_1: intercept
θ_2: coefficient of x
Once we find the best θ_1 and θ_2 values, we get the best fit line. So when we are finally using our model for prediction, it will predict the value of y for the input value of x.

Pros	**Cons**
Linear regression performs exceptionally well for linearly separable data	The assumption of linearity between dependent and independent variables
Easier to implement, interpret and efficient to train	It is often quite prone to noise and overfitting
It handles overfitting pretty well using dimensionally reduction techniques, regularization, and cross-validation	Linear regression is quite sensitive to outliers
One more advantage is the extrapolation beyond a specific data set	It is prone to multicollinearity

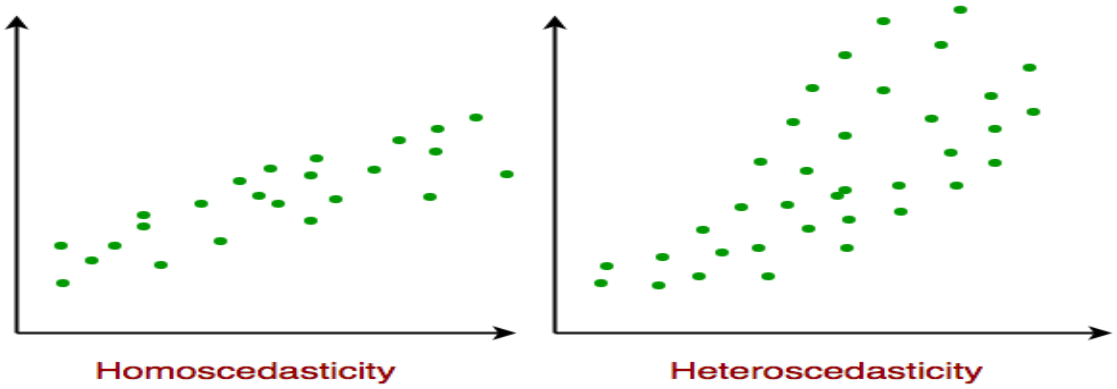
Assumptions:

Given below are the basic assumptions that a linear regression model makes regarding a dataset on which it is applied:

- **Linear relationship**: Relationship between response and feature variables should be linear. The linearity assumption can be tested using scatter plots. As shown below, 1st figure represents linearly related variables whereas variables in the 2nd and 3rd figures are most likely non-linear. So, 1st figure will give better predictions using linear regression.



- **Little or no multi-collinearity**: It is assumed that there is little or no multicollinearity in the data. Multicollinearity occurs when the features (or independent variables) are not independent of each other.
- **Little or no auto-correlation**: Another assumption is that there is little or no autocorrelation in the data. Autocorrelation occurs when the residual errors are not independent of each other.
- **Homoscedasticity**: Homoscedasticity describes a situation in which the error term (that is, the "noise" or random disturbance in the relationship between the independent variables and the dependent variable) is the same across all values of the independent variables. As shown below, figure 1 has homoscedasticity while figure 2 has heteroscedasticity.



Model Building

Steps involved before model building process

Step 1-

Here in this case, we have used a user defined encoder instead of Label encoder or One hot encoder. This is being done before the data split.

The reasons being-

- The categories are in a particular order which are to be manually encoded.
- Considering the data being ordinal, we can't use Label encoding as it will encode based on the categories in Alphabetical order, which is not at all required.
- Even, the creation of dummy variables through one hot encoding is not preferable as it will increase the column nos. and it won't serve the purpose required here.

The categories where Manual Ordinal encoder is being used are also discussed previously, which are as follows- Designation, EducationField

Step 2-

Using one hot encoding to encode categorical variables into numeric/ integer format for the ease in model building.

This is done for both Train and Test datasets, respectively

This includes- Channel, Occupation, Gender, MaritalStatus, Zone, PaymentMethod

Step 3-

Converting or typecasting the previously converted discrete categorical variables into integer format for the ease in model building.

This is done for both Train and Test datasets, respectively

This includes- ExistingProdType, NumberOfPolicy, Complaint, CustCareScore

Steps involved in model building process

Step 4-

Splitting the data into Feature(X) and Target(y) variables respectively

Here in this step, we have opted for a splitting based on the Target column (Agent Bonus), which has been dropped from the Feature variables named 'X'

A separate Target variable is being created using the Target Column and is being named 'y'

Step 5-

Splitting the data into Train and Test variables respectively

Here in this step, we have opted for a splitting based on the Train Test split using a sklearn model selection package which has been used to split the data into 70:30 ratio

- Train = 70 units, Test = 30 units

```
The shape of the training records in the dataseti.e., X_train and y_train are (3164, 18) and (3164, 1)
```

```
The shape of the training records in the dataseti.e., X_test and y_test are (1356, 18) and (1356, 1)
```

Step 6-

Before the Scaling process, we have kept copy of the train- test datasets (for the X_train and X_test)

This is done for both Train and Test datasets, respectively (as x_train_lr,x_test_lr respectively)

Step 7-

We are using Min Max scaling (Normalization) for the Regression problem

The reasons have been stated before

This is done on both Train and Test datasets, respectively

X_Train datatset-Linear Regression (After Scaling)-Table

	Age	CustTenure	Channel	Occupation	EducationField	Gender	ExistingProdType	Designation	NumberOfPolicy	MaritalStatus	MonthlyIncome
2461	0.298507	0.417910	0	0	3	0	4	5	3	0	0.304699
3681	0.865672	0.388060	1	0	6	0	4	4	5	0	0.475685
1309	0.388060	0.119403	1	0	4	0	3	5	1	0	0.014356
4254	0.089552	0.417910	0	0	3	0	4	4	2	0	0.484570
1335	0.179104	0.447761	1	0	4	0	1	5	1	0	0.081116

X_Test datatset- Linear Regression (After Scaling)-Table

	Age	CustTenure	Channel	Occupation	EducationField	Gender	ExistingProdType	Designation	NumberOfPolicy	MaritalStatus	MonthlyIncome
610	0.268657	0.626866	0	0	4	0	3	4	2	0	0.434355
1519	0.537313	0.149254	1	0	6	1	3	1	1	0	1.000000
1620	0.149254	0.388060	1	0	2	0	3	4	1	0	0.173047
2031	0.328358	0.597015	0	0	6	0	3	4	4	0	0.346093
494	0.298507	0.358209	1	0	6	0	3	5	3	1	0.080472

Now we will be discussing about the two Linear regression methods we have opted in this case.

#Method 1-Using Sklearn

The first process being the standard one where the dataset is being applied with the Regression model, which is done with help of the package Sklearn and the different modules as required.

Coefficients of different features - Table

```
The coefficient for Age is 0.18399577350263294
The coefficient for CustTenure is 0.19376820111428134
The coefficient for Channel is -0.003211059962985455
The coefficient for Occupation is 0.1640401250609918
The coefficient for EducationField is 0.00016793344988992
The coefficient for Gender is -0.005806114058411652
The coefficient for ExistingProdType is -0.0038008993969528256
The coefficient for Designation is -0.010544892097017644
The coefficient for NumberOfPolicy is 0.004093172681626175
The coefficient for MaritalStatus is 0.007125750999862693
The coefficient for MonthlyIncome is 0.1673107350848965
The coefficient for Complaint is 0.007431623712150246
The coefficient for ExistingPolicyTenure is 0.10397006000285693
The coefficient for SumAssured is 0.9096524909142808
The coefficient for Zone is -0.005963606392855563
The coefficient for PaymentMethod is -0.0017748333039331206
The coefficient for LastMonthCalls is 0.012997627912221742
The coefficient for CustCareScore is 0.0028054723597570894
```

The value of the intercept is shown below-

```
The intercept for our model is 7.649634854607094
```

Metrices in Linear Regression-Table

Metrices	Train	Test	Comments
MSE	**2.67%**	**2.77%**	Here, both the MSE values for Train and Test datasets are low, which signifies that the Model is better for prediction, as due to lower value, data points are dispersed closely around its central moment
RMSE	**16.35%**	**16.65%**	Here, both the RMSE values for Train and Test datasets are low to moderate, which signifies that the Model is still good for prediction, as due to lower value, spread of the residuals is not too high
MAPE	**1.6%**	**1.64%**	Here, both the MAPE values for Train and Test datasets are quite low, which signifies that the Model is good for prediction, as due to lower value**,** significantly the accuracy of a forecast system increases
MAE	**13.15%**	**13.49%**	Here, both the MAE values for Train and Test datasets are moderately low, which signifies that the Model is good for prediction, as due to lower value**,** significantly the accuracy of the predictive model increases.
R^2 Score	**76.39%**	**75.40%**	Here, both the MAE values for Train and Test datasets are high, which signifies that the Model is overall good for prediction, as due to higher value**,** significantly the independent variables are being more or less able to explain the variance for the target variable

#Method 2-Using Statsmodel

The second process being the different one where the dataset is being applied with the Regression model, which is done with help of the library Stats model.

Scikit does not provide a facility for adjusted R^2. So, we have used statsmodel, a library that gives results similar to what you obtain in R language

This library expects the X and Y to be given in one single data frame.

Stats model Summary Output Table

OLS Regression Results

Dep. Variable:	AgentBonus	R-squared:	0.764
Model:	OLS	Adj. R-squared:	0.763
Method:	Least Squares	F-statistic:	565.3
Date:	Sat, 16 Jul 2022	Prob (F-statistic):	0.00
Time:	12:13:42	Log-Likelihood:	1238.6
No. Observations:	3164	AIC:	-2439.
Df Residuals:	3145	BIC:	-2324.
Df Model:	18		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	7.6496	0.030	252.379	0.000	7.590	7.709
Age	0.1840	0.013	13.814	0.000	0.158	0.210
CustTenure	0.1938	0.013	14.448	0.000	0.167	0.220
Channel	-0.0032	0.006	-0.500	0.617	-0.016	0.009
Occupation	0.1640	0.116	1.408	0.159	-0.064	0.392
EducationField	0.0002	0.002	0.071	0.943	-0.004	0.005
Gender	-0.0058	0.006	-0.975	0.329	-0.017	0.006
ExistingProdType	-0.0038	0.003	-1.216	0.224	-0.010	0.002
Designation	-0.0105	0.005	-2.272	0.023	-0.020	-0.001
NumberOfPolicy	0.0041	0.002	1.970	0.049	1.98e-05	0.008
MaritalStatus	0.0071	0.008	0.937	0.349	-0.008	0.022
MonthlyIncome	0.1673	0.021	7.947	0.000	0.126	0.209
Complaint	0.0074	0.006	1.144	0.253	-0.005	0.020
ExistingPolicyTenure	0.1040	0.010	10.736	0.000	0.085	0.123
SumAssured	0.9097	0.017	52.929	0.000	0.876	0.943
Zone	-0.0060	0.025	-0.239	0.811	-0.055	0.043
PaymentMethod	-0.0018	0.006	-0.291	0.771	-0.014	0.010
LastMonthCalls	0.0130	0.015	0.881	0.378	-0.016	0.042
CustCareScore	0.0028	0.002	1.317	0.188	-0.001	0.007

Omnibus:	0.208	Durbin-Watson:	2.001
Prob(Omnibus):	0.901	Jarque-Bera (JB):	0.242
Skew:	0.016	Prob(JB):	0.886
Kurtosis:	2.972	Cond. No.	348.

Output from VIF

- Variance inflation factor (vif) is a measure of the amount of multicollinearity in a set of multiple regression variables.
- Mathematically, the vif for a regression model variable is equal to the ratio of the overall model variance to the variance of a model that includes only that single independent variable.
- This ratio is calculated for each independent variable. A high vif indicates that the associated independent variable is highly collinear with the other variables in the model.

VIF Values in Linear Regression- Table

```
Age  VIF =  1.32
CustTenure  VIF =  1.31
Channel  VIF =  1.01
Occupation  VIF =  1.01
EducationField  VIF =  1.01
Gender  VIF =  1.01
ExistingProdType  VIF =  1.18
Designation  VIF =  3.2
NumberOfPolicy  VIF =  1.07
MaritalStatus  VIF =  1.01
MonthlyIncome  VIF =  3.36
Complaint  VIF =  1.01
ExistingPolicyTenure  VIF =  1.1
SumAssured  VIF =  1.71
Zone  VIF =  1.01
PaymentMethod  VIF =  1.06
LastMonthCalls  VIF =  1.18
CustCareScore  VIF =  1.01
```

The vif values in this dataset is considerably low, even though presence of multicollinearity within data features are there.

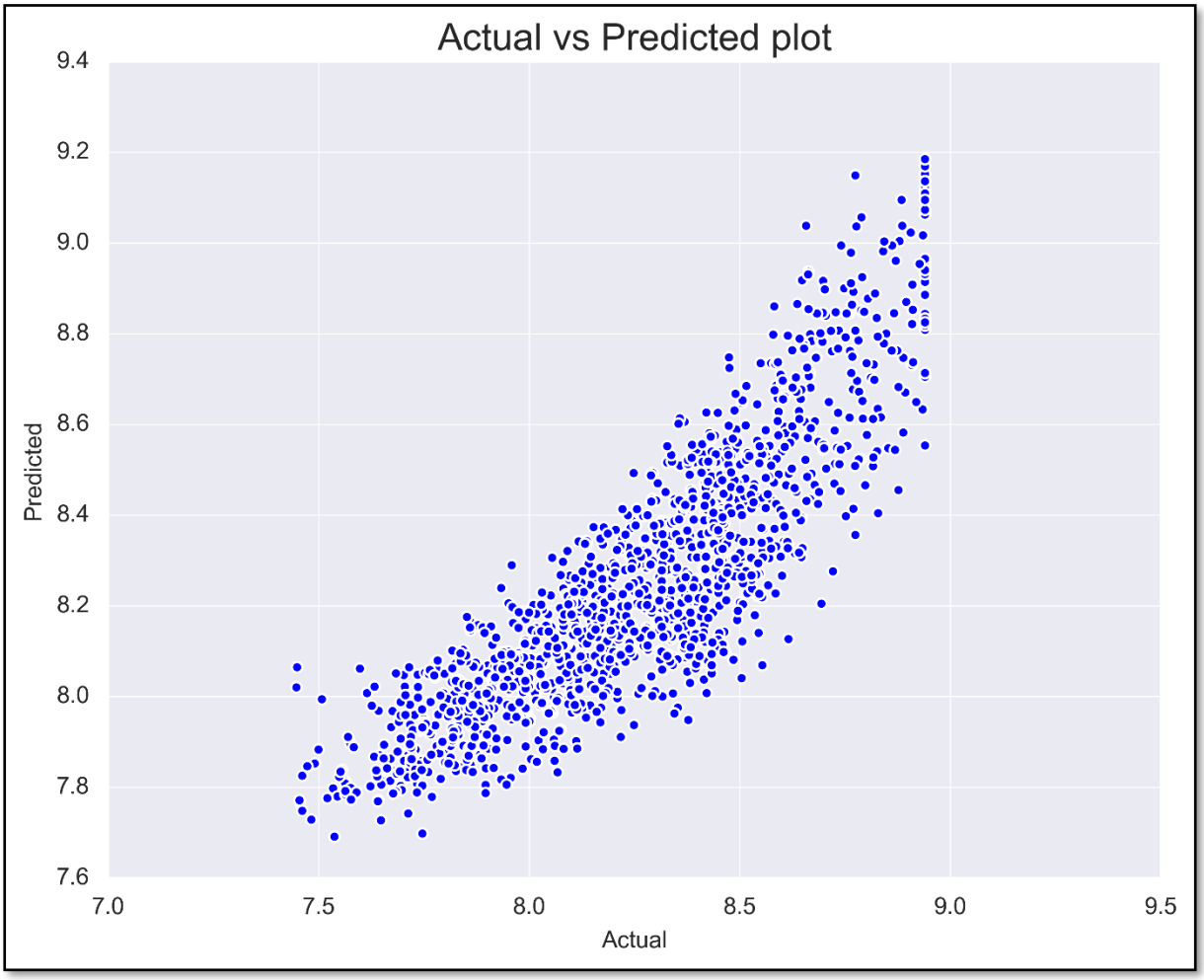
Output in terms of the Actual vs Predicted price plot generated

The output from the qq plot has shown that the predicted and actual price values almost follow a curve line or an area in general, with low-moderate disparity or variance to be precise.

But to presence of multicollinearity, the residual errors are moderate, causing the dispersion of the data, as observed from the graph

Also, it forms a curvilinear line than a linear one.

Scatter plot of Actual vs Predicted price- Linear Regression



Regularization

During the Machine Learning model building, the **Regularization Techniques** is an unavoidable and important step to improve the model prediction and reduce errors. This is also called the **Shrinkage method**. Which we use to add the penalty term to the best fit derived from the trained data, to achieve a *lesser variance* with the tested data and also restricts the influence of predictor variables over the output variable by compressing their coefficients. Thus, this process, controls the complex model to avoid overfitting by reducing the variance (between the train and test data). It is being explained in the following-

Regularization is a technique in machine learning that tries to achieve the generalization of the model. It means that our model works well not only with training or test data, but also with the data it'll receive in the future. In summary, to achieve this, regularization shrinks the weights toward zero to discourage complex models.

In regularization, what we do is normally we keep the same number of features but reduce the magnitude of the coefficients. We can reduce the magnitude of the coefficients by using different types of regression techniques which uses regularization to overcome this problem

There are three main techniques for regularization in linear regression:

- Lasso Regression
- Ridge Regression
- Elastic Net

Pros

- Avoids overfitting a model.
- They do not require unbiased estimators.
- They add just enough bias to make the estimates reasonably reliable approximations to true population values.
- They still perform well in cases of a large multivariate data with the number of predictors (p) larger than the number of observations (n).
- The ridge estimator is preferably good at improving the least-squares estimate when there is multicollinearity.

Cons

- They include all the predictors in the final model.
- They are unable to perform feature selection.
- They shrink the coefficients towards zero.
- They trade the variance for bias.

RIDGE REGRESSION

Ridge regression is a model tuning method that is used to analyse any data that suffers from multicollinearity. This method performs L2 regularization. When the issue of multicollinearity occurs, least-squares are unbiased, and variances are large, this results in predicted values being far away from the actual values.

The cost function for ridge regression:

$$\mathbf{Min(||Y - X(theta)||^2 + \lambda||theta||^2)}$$

Lambda is the penalty term. Λ given here is denoted by an alpha parameter in the ridge function. So, by changing the values of alpha, we are controlling the penalty term. The higher the values of alpha, the bigger is the penalty and therefore the magnitude of coefficients is reduced.

Ridge Regression Models

For any type of regression machine learning model, the usual regression equation forms the base which is written as:

$$\mathbf{Y = XB + e}$$

Where Y is the **dependent variable**, **X** represents the **independent variables**, **B** is the **regression coefficients to be estimated**, and **e** represents the **errors are residuals**.

Once we add the lambda function to this equation, the variance that is not evaluated by the general model is considered. After the data is ready and identified to be part of L2 regularization, there are steps that one can undertake.

Standardization

In ridge regression, the first step is to standardize the variables (both dependent and independent) by subtracting their means and dividing by their standard deviations. This causes a challenge in notation since we must somehow indicate whether the variables in a particular formula are standardized or not. As far as standardization is concerned, all ridge regression calculations are based on standardized variables. When the final regression coefficients are displayed, they are adjusted back into their original scale. However, the ridge trace is on a standardized scale.

Bias and variance trade-off

Bias and variance trade-off is generally complicated when it comes to building ridge regression models on an actual dataset. However, following the general trend which one needs to remember is:

The bias increases as λ increases.

The variance decreases as λ increases.

Assumptions of Ridge Regressions

The assumptions of ridge regression are the same as that of linear regression: linearity, constant variance, and independence. However, as ridge regression does not provide confidence limits, the distribution of errors to be normal need not be assumed.

Pros

- It protects the model from overfitting.
- It does not need unbiased estimators.
- There is only enough bias to make the estimates reasonably reliable approximations to the true population values.
- It performs well when there is a large multivariate data with the number of predictors (p) larger than the number of observations (n).
- The ridge estimator is very effective when it comes to improving the least-squares estimate in situations where there is multicollinearity.
- Model complexity is reduced.

Cons

- It includes all the predictors in the final model.
- It is not capable of performing feature selection.
- It shrinks coefficients towards zero.
- It trades variance for bias.

Model Building

Steps involved before model building process

Step 1-

Here in this case, we have used a user defined encoder instead of Label encoder or One hot encoder. This is being done before the data split.

The reasons being-

- The categories are in a particular order which are to be manually encoded.
- Considering the data being ordinal, we can't use Label encoding as it will encode based on the categories in Alphabetical order, which is not at all required.
- Even, the creation of dummy variables through one hot encoding is not preferable as it will increase the column nos. and it won't serve the purpose required here.

The categories where Manual Ordinal encoder is being used are also discussed previously, which are as follows- Designation, EducationField

Step 2-

Using one hot encoding to encode categorical variables into numeric/ integer format for the ease in model building.

This is done for both Train and Test datasets, respectively

This includes- Channel, Occupation, Gender, MaritalStatus, Zone, PaymentMethod

Step 3-

Converting or typecasting the previously converted discrete categorical variables into integer format for the ease in model building.

This is done for both Train and Test datasets, respectively

This includes- ExistingProdType, NumberOfPolicy, Complaint, CustCareScore

Steps involved in model building process

Step 4-

Splitting the data into Feature(X) and Target(y) variables respectively

Here in this step, we have opted for a splitting based on the Target column (Agent Bonus), which has been dropped from the Feature variables named 'X'

A separate Target variable is being created using the Target Column and is being named 'y'

Step 5-

Splitting the data into Train and Test variables respectively

Here in this step, we have opted for a splitting based on the Train Test split using a sklearn model selection package which has been used to split the data into 70:30 ratio

- Train = 70 units, Test = 30 units

```
The shape of the training records in the dataseti.e., X_train and y_train are (3164, 18) and (3164, 1)
```

```
The shape of the training records in the dataseti.e., X_test and y_test are (1356, 18) and (1356, 1)
```

Step 6-

Before the Scaling process, we have kept copy of the train- test datasets (for the X_train and X_test)

This is done for both Train and Test datasets, respectively (as x_train_rr,x_test_rr respectively)

Step 7-

We are using Standard scaling (Standard) for the Regression problem

The reasons have been stated before

This is done on both Train and Test datasets, respectively

X_Train datatset-Ridge Regression (After Scaling)-Table

	Age	CustTenure	Channel	Occupation	EducationField	Gender	ExistingProdType	Designation	NumberOfPolicy	MaritalStatus	MonthlyIncome
2461	-0.267690	0.204927	0	0	3	0	4	5	3	0	-0.448526
3681	1.988842	0.085165	1	0	6	0	4	4	5	0	0.224800
1309	0.088604	-0.992686	1	0	4	0	3	5	1	0	-1.591863
4254	-1.099044	0.204927	0	0	3	0	4	4	2	0	0.259785
1335	-0.742750	0.324688	1	0	4	0	1	5	1	0	-1.328971

X_Test datatset- Ridge Regression (After Scaling)-Table

	Age	CustTenure	Channel	Occupation	EducationField	Gender	ExistingProdType	Designation	NumberOfPolicy	MaritalStatus	MonthlyIncome
610	-0.386455	1.043256	0	0	4	0	3	4	2	0	0.062046
1519	0.682429	-0.872925	1	0	6	1	3	1	1	0	2.289492
1620	-0.861514	0.085165	1	0	2	0	3	4	1	0	-0.966957
2031	-0.148925	0.923494	0	0	6	0	3	4	4	0	-0.285518
494	-0.267690	-0.034596	1	0	6	0	3	5	3	1	-1.331507

#Method 1-Using Sklearn

The first process being the standard one where the dataset is being applied with the Regression model, which is done with help of the package Sklearn and the different modules as required.

Coefficients of different features - Table

```
The coefficient for Age is 0.04625929979806786
The coefficient for CustTenure is 0.04828912734624076
The coefficient for Channel is -0.0031904823732231426
The coefficient for Occupation is 0.14249671452117232
The coefficient for EducationField is 0.00014702226849726564
The coefficient for Gender is -0.0058262437747308715
The coefficient for ExistingProdType is -0.0037873497395882576
The coefficient for Designation is -0.01055650215662955
The coefficient for NumberOfPolicy is 0.004094866213025317
The coefficient for MaritalStatus is 0.00710601358262026
The coefficient for MonthlyIncome is 0.04247028643471142
The coefficient for Complaint is 0.007474636422637179
The coefficient for ExistingPolicyTenure is 0.03289920212844591
The coefficient for SumAssured is 0.20169468028377147
The coefficient for Zone is -0.005938833011933118
The coefficient for PaymentMethod is -0.0017962903142986844
The coefficient for LastMonthCalls is 0.0027777211302342695
The coefficient for CustCareScore is 0.0028045600423189493
```

The value of the intercept is shown below-

```
The intercept for our model is 8.288591218602464
```

Metrices in Ridge Regression-Table

Metrices	Train	Test	Comments
MSE	**2.67%**	**2.77%**	Here, both the MSE values for Train and Test datasets are low, which signifies that the Model is better for prediction, as due to lower value, data points are dispersed closely around its central moment
RMSE	**16.35%**	**16.65%**	Here, both the RMSE values for Train and Test datasets are low to moderate, which signifies that the Model is still good for prediction, as due to lower value, spread of the residuals is not too high
MAPE	**1.6%**	**1.64%**	Here, both the MAPE values for Train and Test datasets are quite low, which signifies that the Model is good for prediction, as due to lower value**,** significantly the accuracy of a forecast system increases
MAE	**13.15%**	**13.49%**	Here, both the MAE values for Train and Test datasets are moderately low, which signifies that the Model is good for prediction, as due to lower value**,** significantly the accuracy of the predictive model increases.
R^2 Score	**76.39%**	**75.40%**	Here, both the MAE values for Train and Test datasets are moderately high, which signifies that the Model is overall good for prediction, as due to higher value**,** significantly the independent variables are being more or less able to explain the variance for the target variable

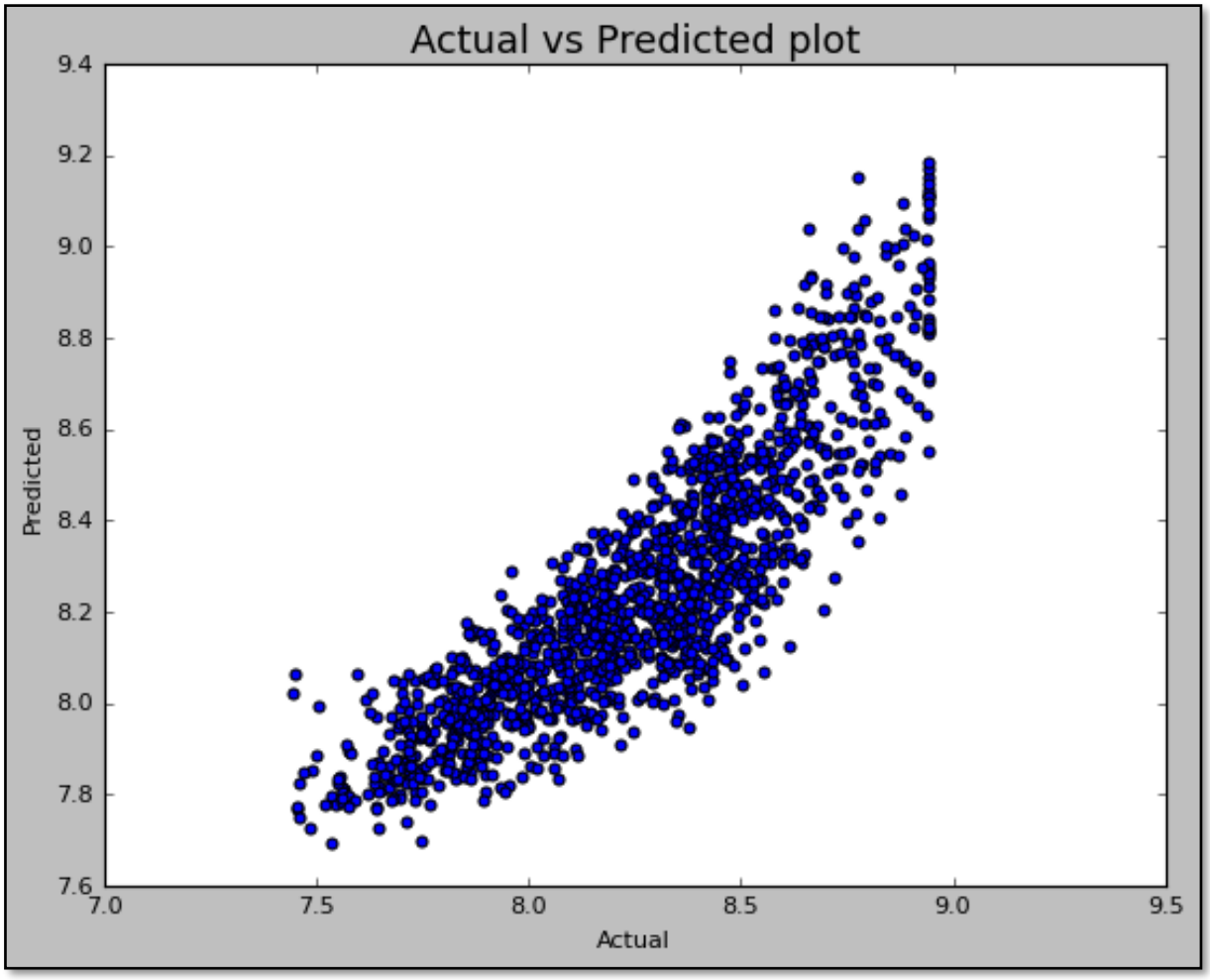
Output in terms of the Actual vs Predicted price plot generated

The output from the qq plot has shown that the predicted and actual price values almost follow a curve line or an area in general, with low-moderate disparity or variance to be precise.

But to presence of multicollinearity, the residual errors are moderate, causing the dispersion of the data, as observed from the graph

Also, it forms a curvilinear line than a linear one.

Scatter plot of Actual vs Predicted price- Ridge Regression



LASSO REGRESSION

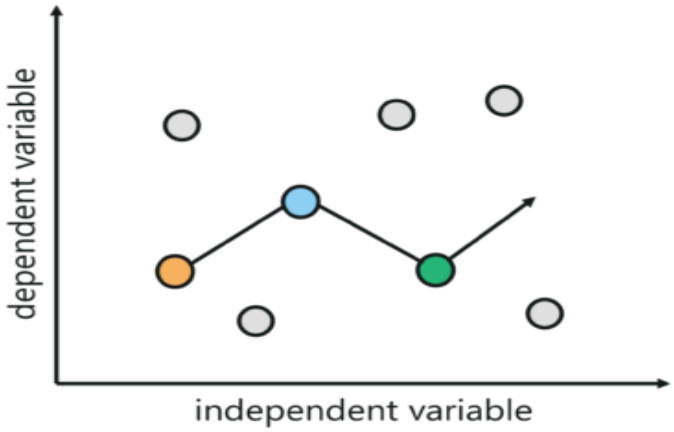
Lasso regression is a regularization technique. It is used over regression methods for a more accurate prediction. This model uses shrinkage. Shrinkage is where data values are shrunk towards a central point as the mean. The lasso procedure encourages simple, sparse models (i.e. models with fewer parameters). This particular type of regression is well-suited for models showing high levels of multicollinearity or when you want to automate certain parts of model selection, like variable selection/parameter elimination. Lasso Regression uses L1 regularization technique (will be discussed later in this article). It is used when we have more features because it automatically performs feature selection.

Lasso Meaning

The word "LASSO" stands for **L**east **A**bsolute **S**hrinkage and **S**election **O**perator. It is a statistical formula for the regularisation of data models and feature selection.

Lasso Regularization Techniques

There are two main regularization techniques, namely Ridge Regression and Lasso Regression. They both differ in the way they assign a penalty to the coefficients. In this blog, we will try to understand more about Lasso Regularization technique.



L1 Regularization

If a regression model uses the L1 Regularization technique, then it is called Lasso Regression. If it used the L2 regularization technique, it's called Ridge Regression. We will study more about these in the later sections.

L1 regularization adds a penalty that is equal to the absolute value of the magnitude of the coefficient. This regularization type can result in sparse models with few coefficients. Some coefficients might become zero and get eliminated from the model. Larger penalties result in coefficient values that are closer to zero (ideal for producing simpler models). On the other hand, L2 regularization does not result in any elimination of sparse models or coefficients. Thus, Lasso Regression is easier to interpret as compared to the Ridge.

Pros

- Select features, by shrinking co-efficient towards zero.
- Avoids over fitting

Cons

- Selected features will be highly biased.
- For n<<p (n-number of data points, p-number of features), LASSO selects at most n features.
- LASSO will select only one feature from a group of correlated features, the selection is arbitrary in nature.
- For different boot strapped data, the feature selected can be very different.
- Prediction performance is worse than Ridge regression.

Model Building

Steps involved before model building process

Step 1-

Here in this case, we have used a user defined encoder instead of Label encoder or One hot encoder. This is being done before the data split.

The reasons being-

- The categories are in a particular order which are to be manually encoded.
- Considering the data being ordinal, we can't use Label encoding as it will encode based on the categories in Alphabetical order, which is not at all required.
- Even, the creation of dummy variables through one hot encoding is not preferable as it will increase the column nos. and it won't serve the purpose required here.

The categories where Manual Ordinal encoder is being used are also discussed previously, which are as follows- Designation, EducationField

Step 2-

Using one hot encoding to encode categorical variables into numeric/ integer format for the ease in model building.

This is done for both Train and Test datasets, respectively

This includes- Channel, Occupation, Gender, MaritalStatus, Zone, PaymentMethod

Step 3-

Converting or typecasting the previously converted discrete categorical variables into integer format for the ease in model building.

This is done for both Train and Test datasets, respectively

This includes- ExistingProdType, NumberOfPolicy, Complaint, CustCareScore

Steps involved in model building process

Step 4-

Splitting the data into Feature(X) and Target(y) variables respectively

Here in this step, we have opted for a splitting based on the Target column (Agent Bonus), which has been dropped from the Feature variables named 'X'

A separate Target variable is being created using the Target Column and is being named 'y'

Step 5-

Splitting the data into Train and Test variables respectively

Here in this step, we have opted for a splitting based on the Train Test split using a sklearn model selection package which has been used to split the data into 70:30 ratio

- Train = 70 units, Test = 30 units

```
The shape of the training records in the dataseti.e., X_train and y_train are (3164, 18) and (3164, 1)
```

```
The shape of the training records in the dataseti.e., X_test and y_test are (1356, 18) and (1356, 1)
```

Step 6-

Before the Scaling process, we have kept copy of the train- test datasets (for the X_train and X_test)

This is done for both Train and Test datasets, respectively (as x_train_lar,x_test_lar respectively)

Step 7-

We are using Standard scaling (Standard) for the Regression problem

The reasons have been stated before

This is done on both Train and Test datasets, respectively

X_Train datatset-Lasso Regression (After Scaling)-Table

	Age	CustTenure	Channel	Occupation	EducationField	Gender	ExistingProdType	Designation	NumberOfPolicy	MaritalStatus	MonthlyIncome
2461	0.298507	0.417910	0	0	3	0	4	5	3	0	0.304699
3681	0.865672	0.388060	1	0	6	0	4	4	5	0	0.475685
1309	0.388060	0.119403	1	0	4	0	3	5	1	0	0.014356
4254	0.089552	0.417910	0	0	3	0	4	4	2	0	0.484570
1335	0.179104	0.447761	1	0	4	0	1	5	1	0	0.081116

X_Test datatset- Lasso Regression (After Scaling)-Table

	Age	CustTenure	Channel	Occupation	EducationField	Gender	ExistingProdType	Designation	NumberOfPolicy	MaritalStatus	MonthlyIncome
610	0.268657	0.626866	0	0	4	0	3	4	2	0	0.434355
1519	0.537313	0.149254	1	0	6	1	3	1	1	0	1.000000
1620	0.149254	0.388060	1	0	2	0	3	4	1	0	0.173047
2031	0.328358	0.597015	0	0	6	0	3	4	4	0	0.346093
494	0.298507	0.358209	1	0	6	0	3	5	3	1	0.080472

#Method 1-Using Sklearn

The first process being the standard one where the dataset is being applied with the Regression model, which is done with help of the package Sklearn and the different modules as required.

Coefficients of different features - Table

```
The coefficient for Age is 0.0
The coefficient for CustTenure is 0.0
The coefficient for Channel is 0.0
The coefficient for Occupation is 0.0
The coefficient for EducationField is 0.0
The coefficient for Gender is 0.0
The coefficient for ExistingProdType is 0.0
The coefficient for Designation is -0.00028674684250143 17
The coefficient for NumberOfPolicy is 0.0
The coefficient for MaritalStatus is 0.0
The coefficient for MonthlyIncome is 0.0
The coefficient for Complaint is 0.0
The coefficient for ExistingPolicyTenure is 0.0
The coefficient for SumAssured is 0.17880516082182668
The coefficient for Zone is -0.0
The coefficient for PaymentMethod is 0.0
The coefficient for LastMonthCalls is 0.0
The coefficient for CustCareScore is 0.0
```

The value of the intercept is shown below-

```
The intercept for our model is 8.256006002744842
```

Metrices in Lasso Regression-Table

Metrices	Train	Test	Comments
MSE	**4.55%**	**4.65%**	Here, both the MSE values for Train and Test datasets are low, which signifies that the Model is better for prediction, as due to lower value, data points are dispersed closely around its central moment
RMSE	**21.33%**	**21.57%**	Here, both the RMSE values for Train and Test datasets are low to moderate, which signifies that the Model is still good for prediction, as due to lower value, spread of the residuals is not too high
MAPE	**2.05%**	**2.09%**	Here, both the MAPE values for Train and Test datasets are quite low, which signifies that the Model is good for prediction, as due to lower value**,** significantly the accuracy of a forecast system increases
MAE	**16.19%**	**17.15%**	Here, both the MAE values for Train and Test datasets are low to moderate, which signifies that the Model is good for prediction, as due to lower value**,** significantly the accuracy of the predictive model increases.
R^2 Score	**59.85%**	**58.71%**	Here, both the MAE values for Train and Test datasets are moderate, which signifies that the Model is overall not so good for prediction, as due to moderate values**,** significantly the independent variables are being more or less able to explain the variance for the target variable

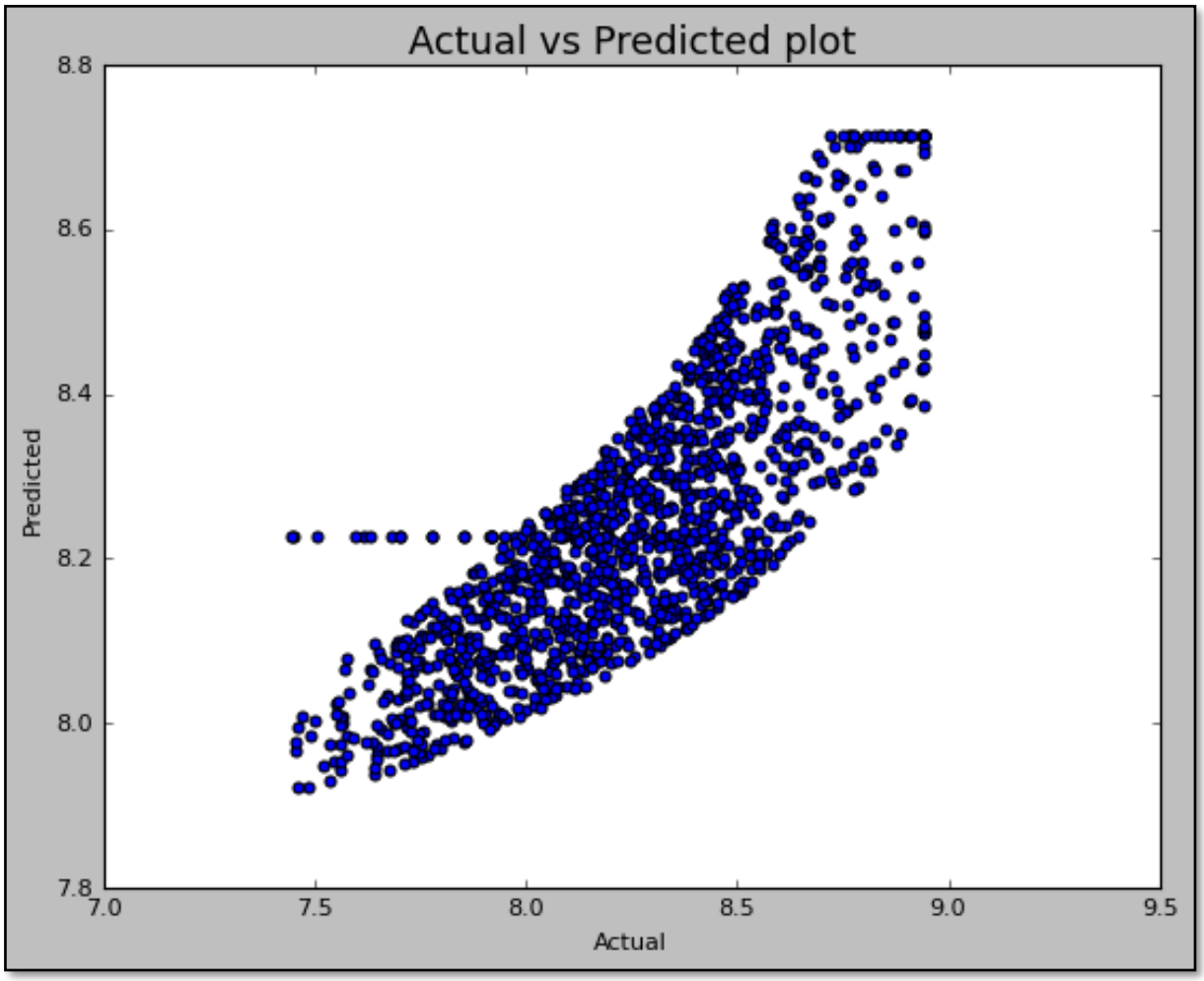
Output in terms of the Actual vs Predicted price plot generated

The output from the qq plot has shown that the predicted and actual price values almost follow a curvilinear patch or an area in general, with low-moderate disparity or variance to be precise.

But to presence of multicollinearity, the residual errors are moderate, causing the dispersion of the data, as observed from the graph

Also, it forms a curvilinear patch than a linear one.

Scatter plot of Actual vs Predicted price- Lasso Regression



DECISION TREE REGRESSOR

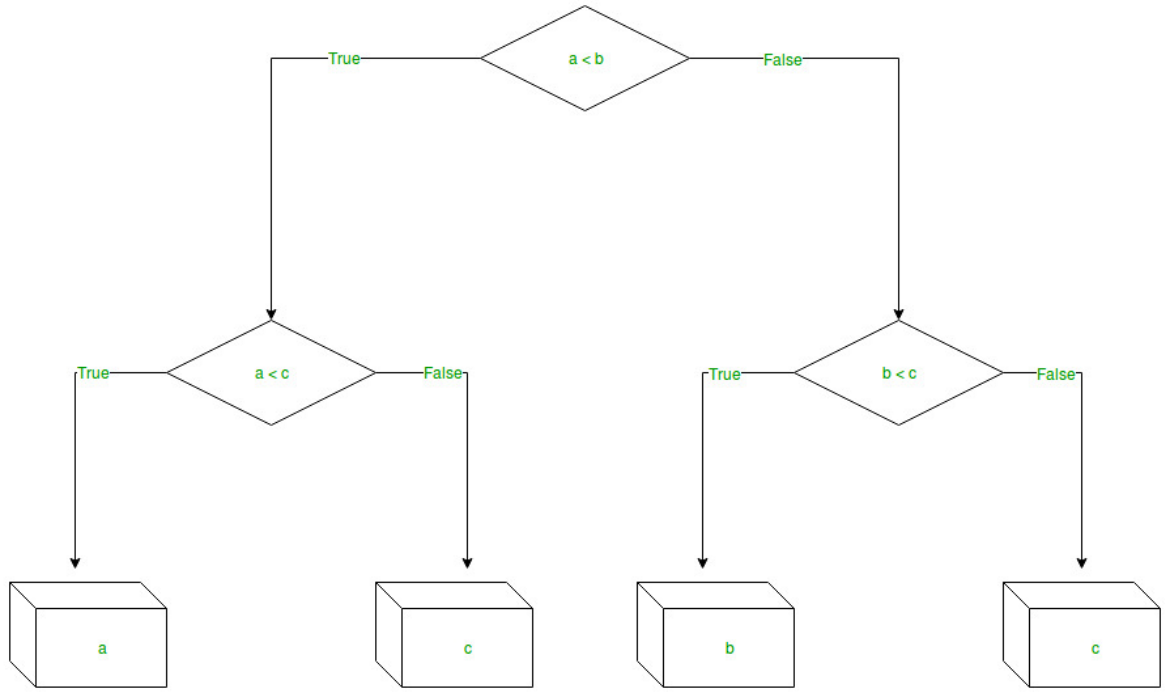
Decision Tree is a decision-making tool that uses a flowchart-like tree structure or is a model of decisions and all of their possible results, including outcomes, input costs, and utility.It builds regression or classification models in the form of a tree structure. It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with **decision nodes** and **leaf nodes**. A decision node (e.g., Outlook) has two or more branches (e.g., Sunny, Overcast and Rainy), each representing values for the attribute tested. Leaf node (e.g., Hours Played) represents a decision on the numerical target. The topmost decision node in a tree which corresponds to the best predictor called **root node**.

Decision trees can handle both categorical and numerical data.

The branches/edges represent the result of the node and the nodes have either:

1. Conditions [Decision Nodes]
2. 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:



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.
Here, continuous values are predicted with the help of a decision tree regression model.

Pros

- The decision tree model can be used for both classification and regression problems, and it is easy to interpret, understand, and visualize.
- The output of a decision tree can also be easily understood.
- Compared with other algorithms, data preparation during pre-processing in a decision tree requires less effort and does not require normalization of data.
- The implementation can also be done without scaling the data.
- A decision tree is one of the quickest ways to identify relationships between variables and the most significant variable.
- New features can also be created for better target variable prediction.
- Decision trees are not largely influenced by outliers or missing values, and it can handle both numerical and categorical variables.
- Since it is a non-parametric method, it has no assumptions about space distributions and classifier structure.

Cons

- Overfitting is one of the practical difficulties for decision tree models. It happens when the learning algorithm continues developing hypotheses that reduce the training set error but at the cost of increasing test set error. But this issue can be resolved by pruning and setting constraints on the model parameters.
- Decision trees cannot be used well with continuous numerical variables.
- A small change in the data tends to cause a big difference in the tree structure, which causes instability.
- Calculations involved can also become complex compared to other algorithms, and it takes a longer time to train the model.
- It is also relatively expensive as the amount of time taken and the complexity levels are greater.

Model Building

Steps involved before model building process

Step 1-

Here in this case, we have used a user defined encoder instead of Label encoder or One hot encoder. This is being done before the data split.

The reasons being-

- The categories are in a particular order which are to be manually encoded.
- Considering the data being ordinal, we can't use Label encoding as it will encode based on the categories in Alphabetical order, which is not at all required.
- Even, the creation of dummy variables through one hot encoding is not preferable as it will increase the column nos. and it won't serve the purpose required here.

The categories where Manual Ordinal encoder is being used are also discussed previously, which are as follows- Designation, EducationField

Step 2-

Using label encoding to encode categorical variables into numeric/ integer format for the ease in model building.

This is done for both Train and Test datasets, respectively

This includes- Channel, Occupation, Gender, MaritalStatus, Zone, PaymentMethod

Step 3-

Converting or typecasting the previously converted discrete categorical variables into integer format for the ease in model building.

This is done for both Train and Test datasets, respectively

This includes- ExistingProdType, NumberOfPolicy, Complaint, CustCareScore

Steps involved in model building process

Step 4-

Splitting the data into Feature(X) and Target(y) variables respectively

Here in this step, we have opted for a splitting based on the Target column (Agent Bonus), which has been dropped from the Feature variables named 'X'

A separate Target variable is being created using the Target Column and is being named 'y'

Step 5-

Splitting the data into Train and Test variables respectively

Here in this step, we have opted for a splitting based on the Train Test split using a sklearn model selection package which has been used to split the data into 70:30 ratio-

Train = 70 units, Test = 30 units

```
The shape of the training records in the dataseti.e., X_train and y_train are (3164, 18) and (3164, 1)
```

```
The shape of the training records in the dataseti.e., X_test and y_test are (1356, 18) and (1356, 1)
```

#Method 1-Using Sklearn- Without Grid Search CV

The first process being the standard one where in the dataset we apply the Regression model, which is done with help of the package Sklearn and the different modules as required.

For this particular method, we have opted for the Decision Tree Regressor and it being there in the Sklearn package

Step 6-

In this step we apply the Decision Tree Regressor model on the Train data to check the overall model building process.

We first apply the whole model on the train dataset and then test the model on the testing dataset.

We haven't used the scaling technique as it is not required in the Decision Tree Regressor

Actual vs Predicted Values- Decision Tree Regressor

	Real Values	Predicted Values for test
0	8.646290	8.651899
1	8.788441	8.921458
2	7.796058	8.218248
3	8.409831	8.417815
4	7.789869	7.638198
...	...	...
1351	8.551788	8.588211
1352	7.774436	8.018296
1353	7.705713	7.999679
1354	7.818430	7.675546
1355	8.425516	8.535033

1356 rows × 2 columns

Metrices in Decision Tree Regressor-Table

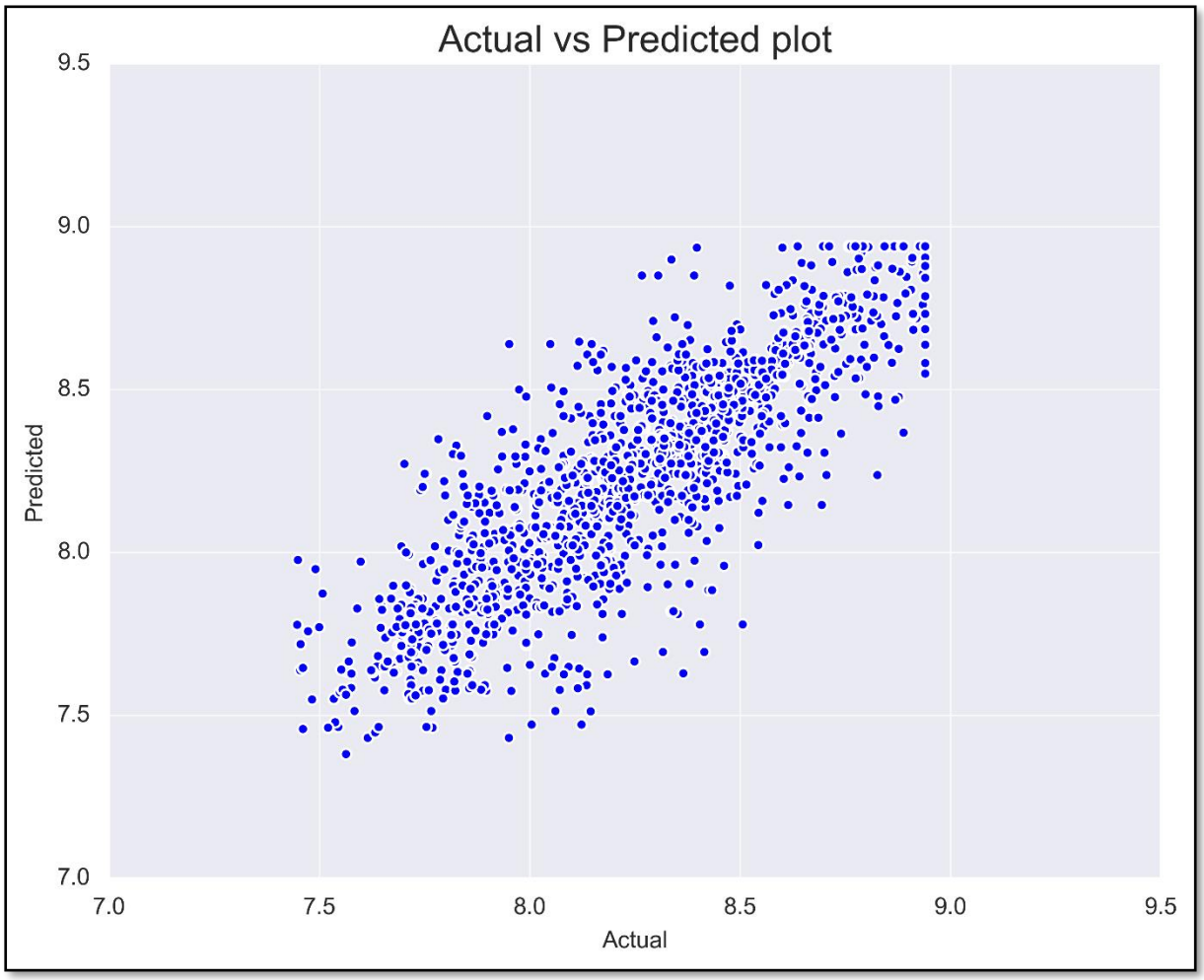
Metrices	Train	Test	Comments
MSE	**1.9945914164880196e-32**	**3.70%**	Here, both the MSE values for Train and Test datasets are low, which signifies that the Model is better for prediction, as due to lower value, data points are dispersed closely around its central moment
RMSE	**1.41230004478086e-16**	**19.23%**	Here, both the RMSE values for Train and Test datasets are low to moderate, which signifies that the Model is still good for prediction, as due to lower value, spread of the residuals is not too high
MAPE	**1.2560766009068325e-18**	**1.68%**	Here, both the MAPE values for Train and Test datasets are quite low, which signifies that the Model is good for prediction, as due to lower value**,** significantly the accuracy of a forecast system increases
MAE	**1.1228551450064794e-17**	**13.82%**	Here, both the MAE values for Train and Test datasets are moderately low, which signifies that the Model is good for prediction, as due to lower value**,** significantly the accuracy of the predictive model increases.
R^2 Score	**100%**	**67.17%**	Here, both the MAE values for Train and Test datasets show a drastic difference, making the Train data overfit for the process. This model is not good from this Metrics POV

Output in terms of the Actual vs Predicted price plot generated

The output from the qq plot has shown that the predicted and actual price values almost follow a curve line or an area in general, with moderate disparity or variance to be precise.

But to presence of multicollinearity, the residual errors are moderately low, causing the dispersion of the data, as observed from the graph.

Scatter plot of Actual vs Predicted price- Decision Tree Regressor

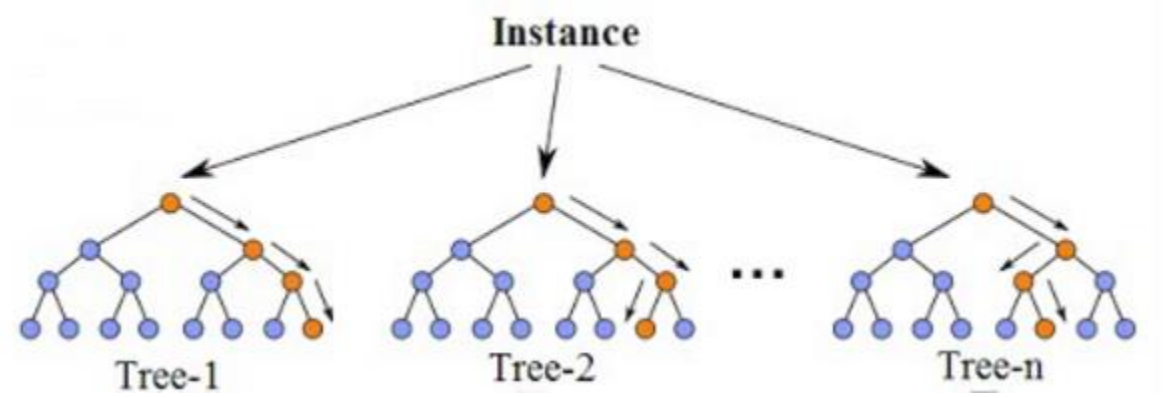


RANDOM FOREST REGRESSOR

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 regression problem, the final output is the mean of all the outputs. This part is called **Aggregation**.

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 is an ensemble of decision trees. This is to say that many trees, constructed in a certain "random" way form a Random Forest.

- Each tree is created from a different sample of rows and at each node, a different sample of features is selected for splitting.
- Each of the trees makes its own individual prediction.
- These predictions are then averaged to produce a single result.



The averaging makes a Random Forest better than a single Decision Tree hence improves its accuracy and reduces overfitting.

A prediction from the Random Forest Regressor is an average of the predictions produced by the trees in the forest.

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

- Design a specific question or data and get the source to determine the required data.

- Make sure the data is in an accessible format else convert it to the required format.
- Specify all noticeable anomalies and missing data points that may be required to achieve the required data.
- Create a machine learning model
- Set the baseline model that you want to achieve
- Train the data machine learning model.
- Provide an insight into the model with test data
- Now compare the performance metrics of both the test data and the predicted data from the model.
- If it doesn't satisfy your expectations, you can try improving your model accordingly or dating your data, or using another data modelling technique.
- At this stage, you interpret the data you have gained and report accordingly.

Pros

- Random forest can decorrelate trees. It picks the training sample and gives each tree a subset of the features (suppose training data was [1,2,3,4,5,6], so one tree will get subset of training data [1,2,3,2,6,6]. Note that size of training data remains same, both datasets have length 6 and that feature '2' and feature '6' are repeated in the randomly sampled training data given to one tree. Each tree predicts according to the features it has. In this case tree 1 only has access to features 1,2,3 and 6 so it can predict based on these features. Some other tree will have access to features 1,4,5 say so it will predict according to those features. If features are highly correlated then that problem can be tackled in random forest.

- **Reduced error:** Random Forest is an ensemble of decision trees. For predicting the outcome of a particular row, random forest takes inputs from all the trees and then predicts the outcome. This ensures that the individual errors of trees are minimized and overall variance and error is reduced.

- **Good Performance on Imbalanced datasets**: It can also handle errors in imbalanced data (one class is majority and other class is minority)

- **Handling of huge amount of data:** It can handle huge amount of data with higher dimensionality of variables.

- **Good handling of missing data:** It can handle missing data very well. So, if there is large amount of missing data in your model, it will give good results.

- **Little impact of outliers:** As the final outcome is taken by consulting many decision trees so certain data points which are outliers will not have a very big impact on Random Forest.

- **No problem of overfitting:** In Random Forest considers only a subset of features, and the final outcome depends on all the trees. So, there is more generalization and less overfitting.
- **Useful to extract feature importance** (we can use it for feature selection)

Cons

- **Features** need to have **some predictive power** else they won't work.
- **Predictions of the trees need to be uncorrelated**.
- **Appears as Black Box:** It is tough to know what is happening. You can at best try different parameters and random seeds to change the outcomes and performance.

Model Building

Steps involved before model building process

Step 1-

Here in this case, we have used a user defined encoder instead of Label encoder or One hot encoder. This is being done before the data split.

The reasons being-

- The categories are in a particular order which are to be manually encoded.
- Considering the data being ordinal, we can't use Label encoding as it will encode based on the categories in Alphabetical order, which is not at all required.
- Even, the creation of dummy variables through one hot encoding is not preferable as it will increase the column nos. and it won't serve the purpose required here.

The categories where Manual Ordinal encoder is being used are also discussed previously, which are as follows- Designation, EducationField

Step 2-

Using label encoding to encode categorical variables into numeric/ integer format for the ease in model building.

This is done for both Train and Test datasets, respectively

This includes- Channel, Occupation, Gender, MaritalStatus, Zone, PaymentMethod

Step 3-

Converting or typecasting the previously converted discrete categorical variables into integer format for the ease in model building.

This is done for both Train and Test datasets, respectively

This includes- ExistingProdType, NumberOfPolicy, Complaint, CustCareScore

Steps involved in model building process

Step 4-

Splitting the data into Feature(X) and Target(y) variables respectively

Here in this step, we have opted for a splitting based on the Target column (Agent Bonus), which has been dropped from the Feature variables named 'X'

A separate Target variable is being created using the Target Column and is being named 'y'

Step 5-

Splitting the data into Train and Test variables respectively

Here in this step, we have opted for a splitting based on the Train Test split using a sklearn model selection package which has been used to split the data into 70:30 ratio

- Train = 70 units, Test = 30 units

```
The shape of the training records in the dataseti.e., X_train and y_train are (3164, 18) and (3164, 1)
```

```
The shape of the training records in the dataseti.e., X_test and y_test are (1356, 18) and (1356, 1)
```

#Method 1-Using Sklearn- Without Grid Search CV

The first process being the standard one where in the dataset we apply the Regression model, which is done with help of the package Sklearn and the different modules as required.

For this particular method, we have opted for the Random Forest Regressor and it being there in the Sklearn package

Step 6-

In this step we apply the Random Forest Regressor model on the Train data to check the overall model building process.

We first apply the whole model on the train dataset and then test the model on the testing dataset.

We haven't used the scaling technique as it is not required in the Random Forest Regressor

Actual vs Predicted Values- Random Forest Regressor

	Real Values	Predicted Values for test
0	8.646290	8.468359
1	8.788441	8.887640
2	7.796058	8.006456
3	8.409831	8.513813
4	7.789869	7.644339
...	...	...
1351	8.551788	8.551834
1352	7.774436	7.995415
1353	7.705713	8.018257
1354	7.818430	7.852888
1355	8.425516	8.482844

1356 rows × 2 columns

Metrices in Random Forest Regressor-Table

Metrices	Train	Test	Comments
MSE	**0.24%**	**1.79%**	Here, both the MSE values for Train and Test datasets are low, which signifies that the Model is better for prediction, as due to lower value, data points are dispersed closely around its central moment
RMSE	**4.93%**	**13.40%**	Here, both the RMSE values for Train and Test datasets are low to moderate, which signifies that the Model is still good for prediction, as due to lower value, spread of the residuals is not too high in the train dataset, but there has been a drastic difference between the train and test data values
MAPE	**0.46%**	**1.25%**	Here, both the MAPE values for Train and Test datasets are quite low, which signifies that the Model is good for prediction, as due to lower value**,** significantly the accuracy of a forecast system increases
MAE	**3.77%**	**10.30%**	Here, both the MAE values for Train and Test datasets are quite low, which signifies that the Model is good for prediction, as due to lower value**,** significantly the accuracy of the predictive model increases. But there is a significant increase in the Test data value as compared to the Train dataset
R^2 Score	**97.84%**	**84.04%**	Here, both the MAE values for Train and Test datasets are quite high, which signifies that the Model is good for prediction, as due to higher value**,** significantly the independent variables are being able to explain the variance for the target variable

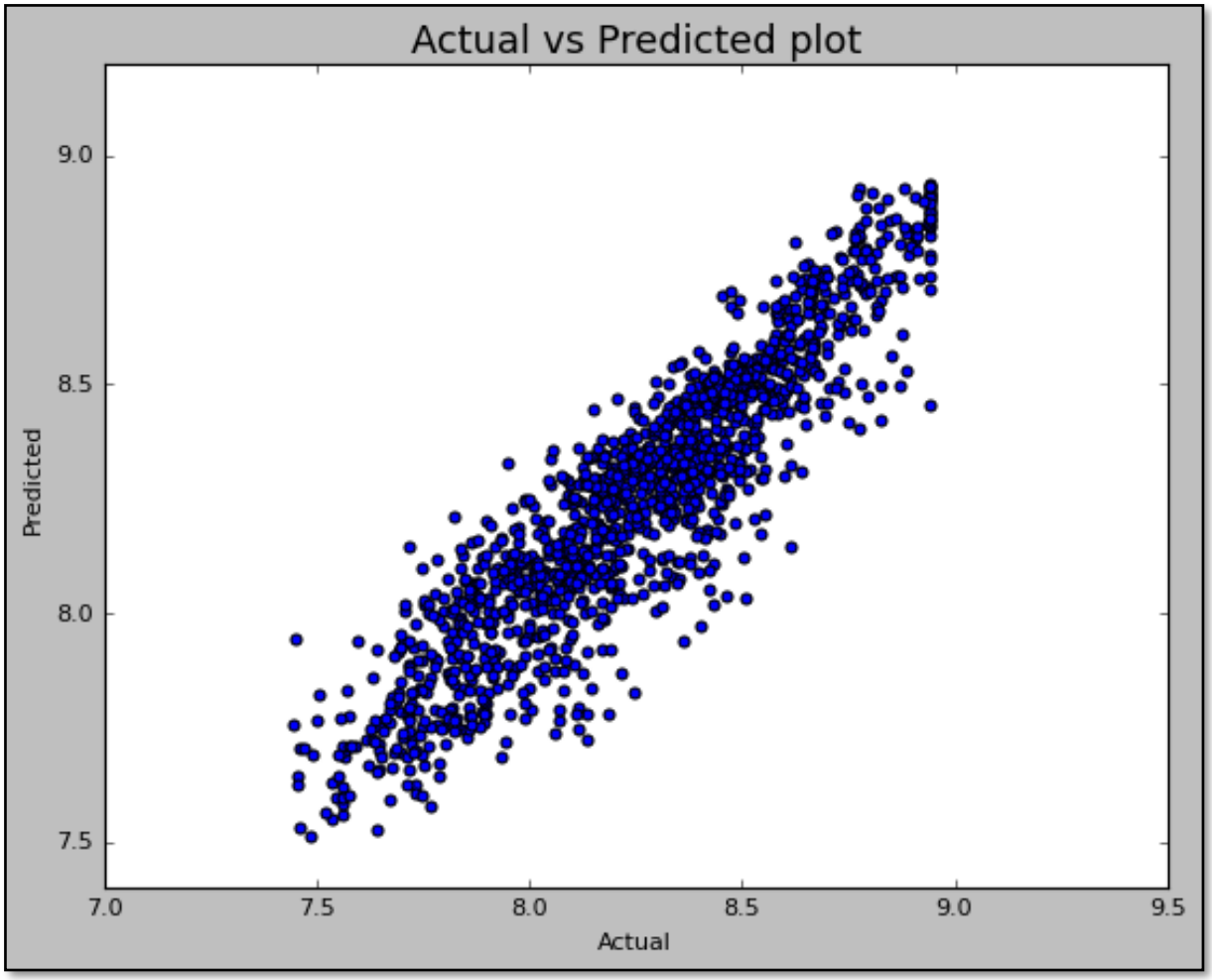
Output in terms of the Actual vs Predicted price plot generated

The output from the qq plot has shown that the predicted and actual price values almost follow a curve line or an area in general, with low-moderate disparity or variance to be precise.

But to presence of multicollinearity, the residual errors are low, causing the dispersion of the data, as observed from the graph.

Also, it shows a more or less linear trend for the residuals

Scatter plot of Actual vs Predicted price- Random Forest Regressor



K-NEAREST NEIGHBOUR

KNN regression is a non-parametric method that, in an intuitive manner, approximates the association between independent variables and the continuous outcome by averaging the observations in the same *neighbourhood*. The size of the neighbourhood needs to be set by the analyst or can be chosen using cross-validation (we will see this later) to select the size that minimises the mean-squared error.
While the method is quite appealing, it quickly becomes impractical when the dimension increases, i.e., when there are many independent variables.

Uses-

KNN regression tries to predict the **value** of the output variable by using a local average.

Pros

- **Simple** to understand and implement
- **No assumption about data** (for e.g. in case of linear regression we assume dependent variable and independent variables are linearly related, in Naïve Bayes we assume features are independent of each other etc., but k-NN makes no assumptions about data)
- **Constantly evolving** model: When it is exposed to new data, it changes to accommodate the new data points.
- **Multi-class** problems can also be solved.
- **One Hyper Parameter:** K-NN might take some time while selecting the first hyper parameter but after that rest of the parameters are aligned to it.

Cons

- **Slow** for large datasets.
- **Curse of dimensionality**: Does not work very well on datasets with large number of features.
- **Scaling** of data absolute must.
- **Does not work well on Imbalanced data.** So before using k-NN either undersample majority class or oversample minority class and have a balanced dataset.
- Sensitive to **outliers**.
- Can't deal well with **missing values**

Applications:

You can use it for any classification problem when dataset is smaller, and has lesser number of features so that computation time taken by k-NN is less. If you do not know the shape of the data and the way output and inputs are related (whether classes can be separated by a line or ellipse or parabola etc.), then you can use k-NN.

Model Building

Steps involved before model building process

Step 1-

Here in this case, we have used a user defined encoder instead of Label encoder or One hot encoder. This is being done before the data split.

The reasons being-

- The categories are in a particular order which are to be manually encoded.
- Considering the data being ordinal, we can't use Label encoding as it will encode based on the categories in Alphabetical order, which is not at all required.
- Even, the creation of dummy variables through one hot encoding is not preferable as it will increase the column nos. and it won't serve the purpose required here.

The categories where Manual Ordinal encoder is being used are also discussed previously, which are as follows- Designation, EducationField

Step 2-

Using label encoding to encode categorical variables into numeric/ integer format for the ease in model building.

This is done for both Train and Test datasets, respectively

This includes- Channel, Occupation, Gender, MaritalStatus, Zone, PaymentMethod

Step 3-

Converting or typecasting the previously converted discrete categorical variables into integer format for the ease in model building.

This is done for both Train and Test datasets, respectively

This includes- ExistingProdType, NumberOfPolicy, Complaint, CustCareScore

Steps involved in model building process

Step 4-

Splitting the data into Feature(X) and Target(y) variables respectively

Here in this step, we have opted for a splitting based on the Target column (Agent Bonus), which has been dropped from the Feature variables named 'X'

A separate Target variable is being created using the Target Column and is being named 'y'

Step 5-

Splitting the data into Train and Test variables respectively

Here in this step, we have opted for a splitting based on the Train Test split using a sklearn model selection package which has been used to split the data into 70:30 ratio

- Train = 70 units, Test = 30 units

```
The shape of the training records in the dataseti.e., X_train and y_train are (3164, 18) and (3164, 1)
```

```
The shape of the training records in the dataseti.e., X_test and y_test are (1356, 18) and (1356, 1)
```

#Method 1-Using Sklearn- Without Grid Search CV

The first process being the standard one where in the dataset we apply the Regression model, which is done with help of the package Sklearn and the different modules as required.

For this particular method, we have opted for the KNN Regressor and it being there in the Sklearn package

Step 6-

Before the Scaling process, we have kept copy of the train- test datasets (for the X_train and X_test)

This is done for both Train and Test datasets, respectively (as x_train_knn,x_test_knn respectively)

Step 7-

We are using Min Max scaling (Normalization) for the Regression problem

The reasons have been stated before

This is done on both Train and Test datasets, respectively

X_Train datatset-KNN(After Scaling)-Table

	Age	CustTenure	Channel	Occupation	EducationField	Gender	ExistingProdType	Designation	NumberOfPolicy	MaritalStatus	MonthlyIncome
2461	0.298507	0.417910	2	1	3	1	4	5	3	2	0.304699
3681	0.865672	0.388060	0	3	6	1	4	4	5	1	0.475685
1309	0.388060	0.119403	0	2	4	1	3	5	1	2	0.014356
4254	0.089552	0.417910	1	1	3	1	4	4	2	1	0.484570
1335	0.179104	0.447761	0	2	4	1	1	5	1	1	0.081116

X_Test datatset-KNN(After Scaling)-Table

	Age	CustTenure	Channel	Occupation	EducationField	Gender	ExistingProdType	Designation	NumberOfPolicy	MaritalStatus	MonthlyIncome
610	0.268657	0.626866	2	2	4	1	3	4	2	2	0.434355
1519	0.537313	0.149254	0	3	6	0	3	1	1	1	1.000000
1620	0.149254	0.388060	0	2	2	1	3	4	1	1	0.173047
2031	0.328358	0.597015	2	3	6	1	3	4	4	1	0.346093
494	0.298507	0.358209	0	3	6	1	3	5	3	0	0.080472

Step 8-

In this step we apply the KNN Regressor model on the Train data to check the overall model building process.

We first apply the whole model on the train dataset and then test the model on the testing dataset.

Actual vs Predicted Values-KNN

	Real Values	Predicted Values for test
0	8.646290	8.336278
1	8.788441	8.780996
2	7.796058	8.063104
3	8.409831	8.478221
4	7.789869	8.128210
...	...	...
1351	8.551788	8.621813
1352	7.774436	8.077924
1353	7.705713	7.960719
1354	7.818430	8.082111
1355	8.425516	8.552733

1356 rows × 2 columns

Metrices in KNN Regressor-Table

Metrices	Train	Test	Comments
MSE	**5.43%**	**7.98%**	Here, both the MSE values for Train and Test datasets are low, which signifies that the Model is better for prediction, as due to lower value, data points are dispersed closely around its central moment
RMSE	**23.30%**	**28.25%**	Here, both the RMSE values for Train and Test datasets are low to moderate, which signifies that the Model is still good for prediction, as due to lower value, spread of the residuals is not too high
MAPE	**2.34%**	**2.87%**	Here, both the MAPE values for Train and Test datasets are quite low, which signifies that the Model is good for prediction, as due to lower value**,** significantly the accuracy of a forecast system increases
MAE	**19.27%**	**23.61%**	Here, both the MAE values for Train and Test datasets are moderately low, which signifies that the Model is good for prediction, as due to lower value**,** significantly the accuracy of the predictive model increases.
R^2 Score	**52.09%**	**29.20%**	Here, both the MAE values for Train and Test datasets are quite low, which signifies that the Model is too bad for prediction, as due to lower value**,** significantly the independent variables are not being able to explain the variance for the target variable as per this metrics is concerned

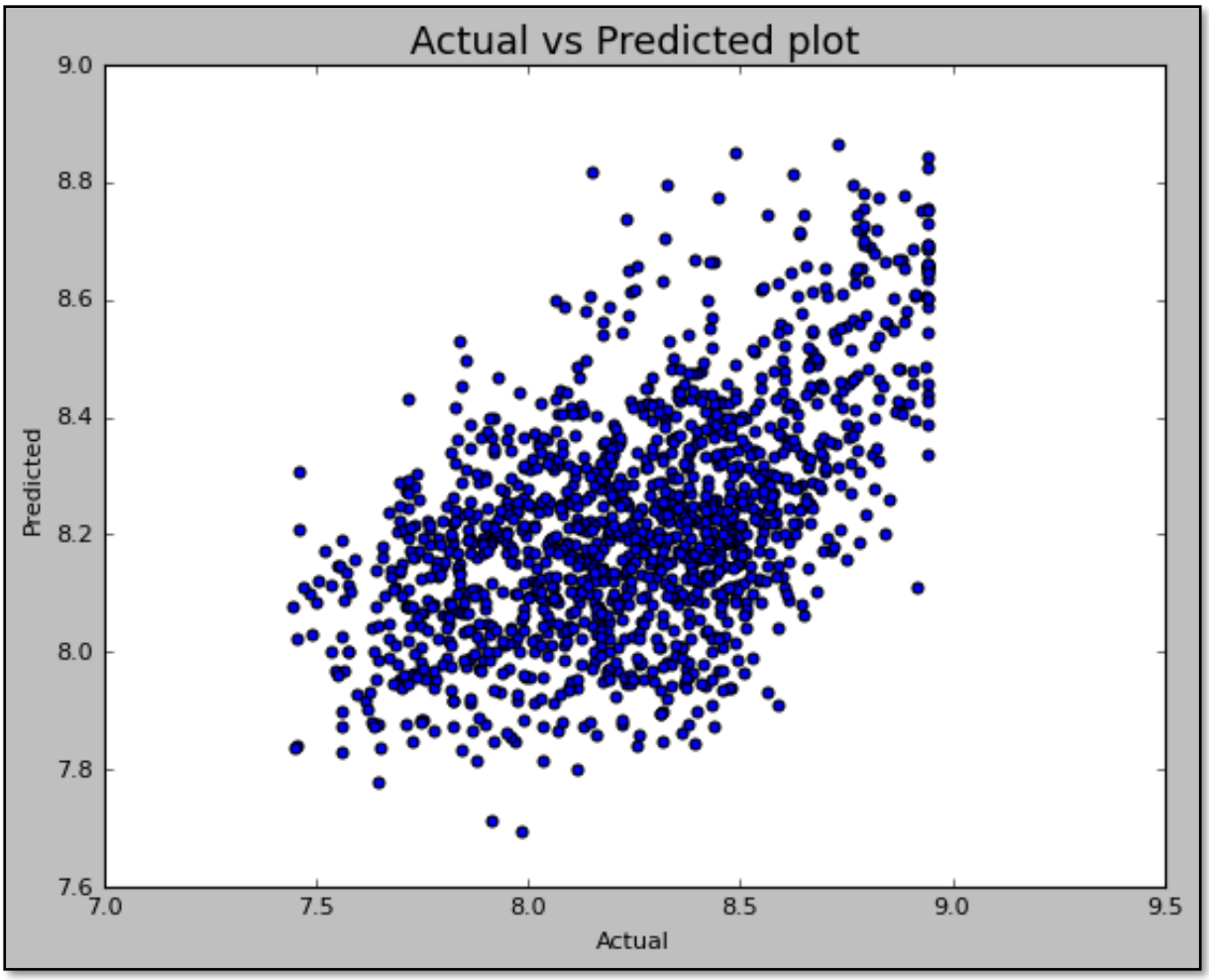
Output in terms of the Actual vs Predicted price plot generated

The output from the qq plot has shown that the predicted and actual price values almost form a clutter due to high disparity

But to presence of multicollinearity, the residual errors are high, causing the dispersion of the data, as observed from the graph.

We are unable to find a trend from this graph

Scatter plot of Actual vs Predicted price-KNN



GRADIENT BOOSTING METHOD

Gradient boosting Regression calculates the difference between the current prediction and the known correct target value.

This difference is called residual. After that Gradient boosting Regression trains a weak model that maps features to that residual. This residual predicted by a weak model is added to the existing model input and thus this process nudges the model towards the correct target. Repeating this step again and again improves the overall model prediction.

Also, it should be noted that Gradient boosting regression is used to predict continuous values like house price, while Gradient Boosting Classification is used for predicting classes like whether a patient has a particular disease or not.

Gradient Boosting parameters

Let us discuss few important parameters used in Gradient Boosting Regression. These are the parameters we may like to tune for getting the best output from our algorithm implementation.

Number of Estimators: It is denoted as n_estimators. The default value of this parameter is 100.Number of estimators is basically the number of boosting stages to be performed by the model. In other words, number of estimators denotes the number of trees in the forest. More number of trees helps in learning the data better. On the other hand, a greater number of trees can result in higher training time. Hence, we need to find the right and balanced value of n_estimators for optimal performance.

Maximum Depth: It is denoted as max_depth. The default value of max_depth is 3 and it is an optional parameter. The maximum depth is the depth of the decision tree estimator in the gradient boosting regressor. We need to find the optimum value of this hyperparameter for best performance. As an example, the best value of this parameter may depend on the input variables.

Learning Rate: It is denoted as learning rate. The default value of learning rate is 0.1 and it is an optional parameter. The learning rate is a hyper-parameter in gradient boosting regressor algorithm that determines the step size at each iteration while moving toward a minimum of a loss function.

Criterion: It is denoted as criterion. The default value of criterion is Friedman_mse and it is an optional parameter. The criterion is used to measure the quality of a split for decision tree. MSE stands for mean squared error.

Loss: It is denoted as loss. The default value of loss is ls and it is an optional parameter. This parameter indicates loss function to be optimized. There are various loss functions like ls which stands for least squares regression. Least absolute deviation abbreviated as lad is another loss function. Huber a third loss function is a combination of least squares regression and least absolute deviation.

Subsample: It is denoted as subsample. The default value of subsample is 1.0 and it is an optional parameter.
Subsample is fraction of samples used for fitting the individual tree learners. If subsample is smaller than 1.0 this leads to a reduction of variance and an increase in bias.

Number of Iteration no change: It is denoted by n_iter_no_change. The default value of subsample is None and it is an optional parameter.
This parameter is used to decide whether early stopping is used to terminate training when validation score is not improving with further iteration.
If this parameter is enabled, it will set aside validation_fraction size of the training data as validation and terminate training when validation score is not improving.

Pros

- **Better accuracy:** Gradient Boosting Regression generally provides better accuracy. When we compare the accuracy of GBR with other regression techniques like Linear Regression, GBR is mostly winner all the time. This is why GBR is being used in most of the online hackathon and competitions.
- **Less pre-processing:** As we know that data pre-processing is one of the vital steps in machine learning workflow, and if we do not do it properly then it affects our model accuracy. However, Gradient Boosting Regression requires minimal data pre-processing, which helps us in implementing this model faster with lesser complexity. Though pre-processing is not mandatory here we should note that we can improve model performance by spending time in pre-processing the data.
- **Higher flexibility:** Gradient Boosting Regression provides can be used with many hyper-parameter and loss functions. This makes the model highly flexible and it can be used to solve a wide variety of problems.
- **Missing data:** Missing data is one of the issues while training a model. Gradient Boosting Regression handles the missing data on its own and does not require us to handle it explicitly. This is clearly a great win over other similar algorithms. In this algorithm the missing values are treated as containing information. Thus, during tree building, splitting decisions for node are decided by minimizing the loss function and treating missing values as a separate category that can go either left or right.

Cons

Let's now address some of the challenges faced when using gradient boosted trees:

- prone to overfitting: this can be solved by applying L1 and L2 regularization penalties. You can try a low learning rate as well;
- models can be computationally expensive and take a long time to train, especially on CPUs;
- hard to interpret the final models.

Model Building

Steps involved before model building process

Step 1-

Here in this case, we have used a user defined encoder instead of Label encoder or One hot encoder. This is being done before the data split.

The reasons being-

- The categories are in a particular order which are to be manually encoded.
- Considering the data being ordinal, we can't use Label encoding as it will encode based on the categories in Alphabetical order, which is not at all required.
- Even, the creation of dummy variables through one hot encoding is not preferable as it will increase the column nos. and it won't serve the purpose required here.

The categories where Manual Ordinal encoder is being used are also discussed previously, which are as follows- Designation, EducationField

Step 2-

Using label encoding to encode categorical variables into numeric/ integer format for the ease in model building.

This is done for both Train and Test datasets, respectively

This includes- Channel, Occupation, Gender, MaritalStatus, Zone, PaymentMethod

Step 3-

Converting or typecasting the previously converted discrete categorical variables into integer format for the ease in model building.

This is done for both Train and Test datasets, respectively

This includes- ExistingProdType, NumberOfPolicy, Complaint, CustCareScore

Steps involved in model building process

Step 4-

Splitting the data into Feature(X) and Target(y) variables respectively

Here in this step, we have opted for a splitting based on the Target column (Agent Bonus), which has been dropped from the Feature variables named 'X'

A separate Target variable is being created using the Target Column and is being named 'y'

Step 5-

Splitting the data into Train and Test variables respectively

Here in this step, we have opted for a splitting based on the Train Test split using a sklearn model selection package which has been used to split the data into 70:30 ratio

- Train = 70 units, Test = 30 units

```
The shape of the training records in the dataseti.e., X_train and y_train are (3164, 18) and (3164, 1)
```

```
The shape of the training records in the dataseti.e., X_test and y_test are (1356, 18) and (1356, 1)
```

#Method 1-Using Sklearn- Without Grid Search CV

The first process being the standard one where in the dataset we apply the Regression model, which is done with help of the package Sklearn and the different modules as required.

For this particular method, we have opted for the Gradient Boost Regressor and it being there in the Sklearn package

Step 6-

In this step we apply the Gradient Boost Regressor model on the Train data to check the overall model building process.

We first apply the whole model on the train dataset and then test the model on the testing dataset.

We haven't used the scaling technique as it is not required in the Gradient Boost Regressor

Actual vs Predicted Values- Gradient Boost Regressor

	Real Values	Predicted Values for test
0	8.646290	8.447597
1	8.788441	8.821693
2	7.796058	8.018569
3	8.409831	8.496856
4	7.789869	7.689449
...	...	...
1351	8.551788	8.495005
1352	7.774436	7.975746
1353	7.705713	7.979989
1354	7.818430	7.823729
1355	8.425516	8.535180

1356 rows × 2 columns

Metrices in Gradient Boost Regressor-Table

Metrices	Train	Test	Comments
MSE	**1.52%**	**1.88%**	Here, both the MSE values for Train and Test datasets are quite low, which signifies that the Model is better for prediction, as due to lower value, data points are dispersed closely around its central moment
RMSE	**12.35%**	**13.72%**	Here, both the RMSE values for Train and Test datasets are low to moderate, which signifies that the Model is still good for prediction, as due to lower value, spread of the residuals is not too high
MAPE	**1.18%**	**1.32%**	Here, both the MAPE values for Train and Test datasets are quite low, which signifies that the Model is good for prediction, as due to lower value**,** significantly the accuracy of a forecast system increases
MAE	**9.73%**	**10.83%**	Here, both the MAE values for Train and Test datasets are moderately low, which signifies that the Model is good for prediction, as due to lower value**,** significantly the accuracy of the predictive model increases.
R^2 Score	**86.52%**	**83.28%**	Here, both the MAE values for Train and Test datasets are quite high, which signifies that the Model is overall good for prediction, as due to higher value**,** significantly the independent variables are being able to explain the variance for the target variable

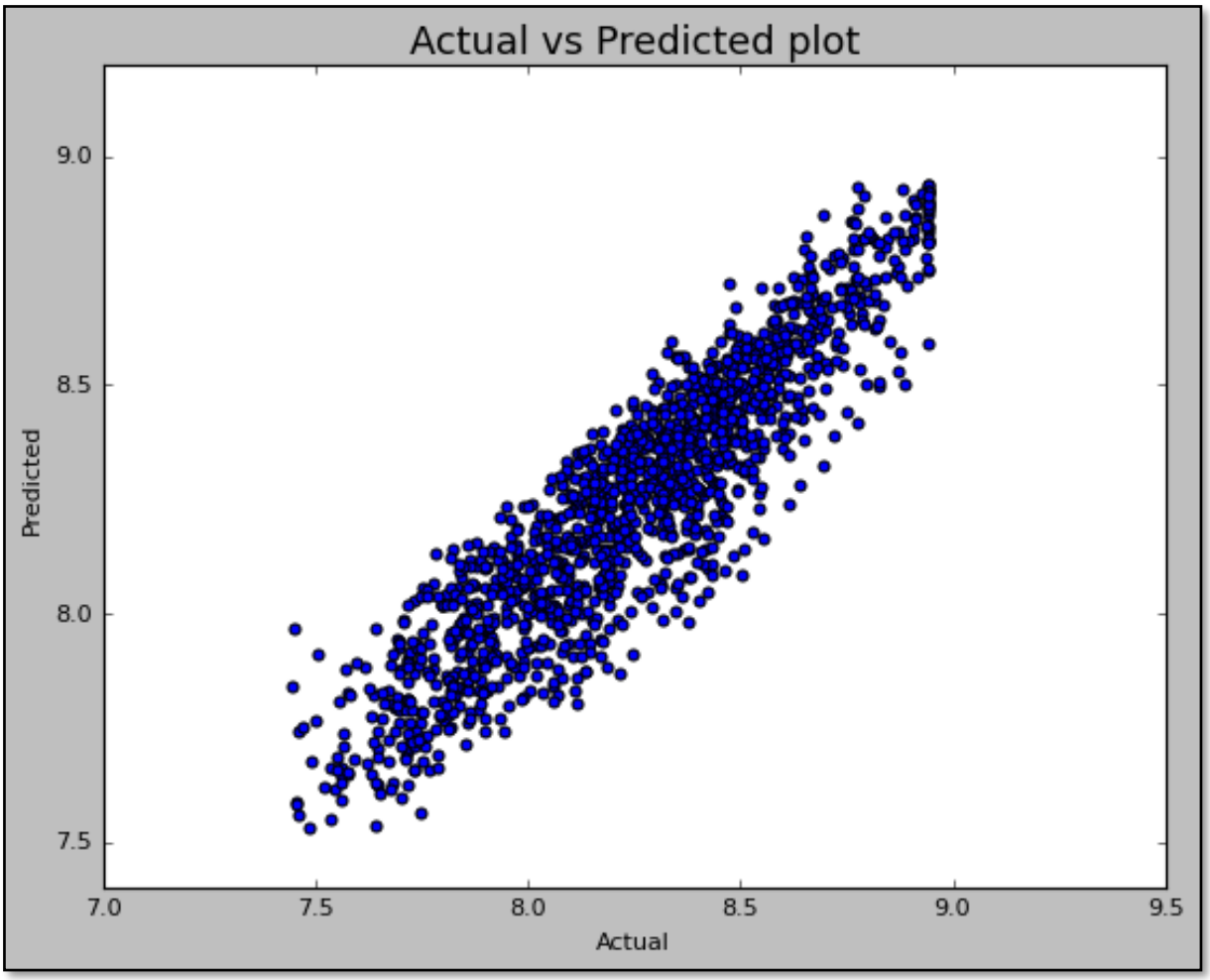
Output in terms of the Actual vs Predicted price plot generated

The output from the qq plot has shown that the predicted and actual price values almost follow a curve line or an area in general, with low-moderate disparity or variance to be precise.

But to presence of multicollinearity, the residual errors are moderately low, causing the dispersion of the data, as observed from the graph.

Also, it shows a more or less linear trend for the residuals

Scatter plot of Actual vs Predicted price- Gradient Boost Regressor



XGBOOSTING METHOD

The results of the regression problems are continuous or real values. Some commonly used regression algorithms are Linear Regression and Decision Trees. There are several metrics involved in regression like root-mean-squared error (RMSE) and mean-squared-error (MAE). These are some key members of XGBoost models, each plays an important role.

- **RMSE:** It is the square root of mean squared error (MSE).
- **MAE:** It is an absolute sum of actual and predicted differences, but it lacks mathematically, that's why it is rarely used, as compared to other metrics.

XGBoost is a powerful approach for building supervised regression models. The validity of this statement can be inferred by knowing about its (XGBoost) objective function and base learners. The objective function contains loss function and a regularization term. It tells about the difference between actual values and predicted values, i.e how far the model results are from the real values. The most common loss functions in XGBoost for regression problems is linear, and that for binary classification is logistics. Ensemble learning involves training and combining individual models (known as base learners) to get a single prediction, and XGBoost is one of the Ensemble learning methods. XGBoost expects to have the base learners which are uniformly bad at the remainder so that when all the predictions are combined, bad predictions cancel out and better one sums up to form final good predictions.

Pros

- **Less feature engineering required** (No need for scaling, normalizing data, can also handle missing values well)
- **Feature importance** can be found out(it shows importance of each feature, can be used for feature selection)
- **Fast** to interpret
- **Outliers** have minimal impact.
- **Handles large sized datasets** well.
- **Good Execution** speed
- **Good model performance** (wins most of the Kaggle competitions)
- **Less prone to overfitting**

Cons

- **Difficult interpretation,** visualization tough
- **Overfitting** possible if parameters not tuned properly.
- **Harder to tune** as there are too many hyperparameters.

Model Building

Steps involved before model building process

Step 1-

Here in this case, we have used a user defined encoder instead of Label encoder or One hot encoder. This is being done before the data split.

The reasons being-

- The categories are in a particular order which are to be manually encoded.
- Considering the data being ordinal, we can't use Label encoding as it will encode based on the categories in Alphabetical order, which is not at all required.
- Even, the creation of dummy variables through one hot encoding is not preferable as it will increase the column nos. and it won't serve the purpose required here.

The categories where Manual Ordinal encoder is being used are also discussed previously, which are as follows- Designation, EducationField

Step 2-

Using label encoding to encode categorical variables into numeric/ integer format for the ease in model building.

This is done for both Train and Test datasets, respectively

This includes- Channel, Occupation, Gender, MaritalStatus, Zone, PaymentMethod

Step 3-

Converting or typecasting the previously converted discrete categorical variables into integer format for the ease in model building.

This is done for both Train and Test datasets, respectively

This includes- ExistingProdType, NumberOfPolicy, Complaint, CustCareScore

Steps involved in model building process

Step 4-

Splitting the data into Feature(X) and Target(y) variables respectively

Here in this step, we have opted for a splitting based on the Target column (Agent Bonus), which has been dropped from the Feature variables named 'X'

A separate Target variable is being created using the Target Column and is being named 'y'

Step 5-

Splitting the data into Train and Test variables respectively

Here in this step, we have opted for a splitting based on the Train Test split using a sklearn model selection package which has been used to split the data into 70:30 ratio

- Train = 70 units, Test = 30 units

```
The shape of the training records in the dataseti.e., X_train and y_train are (3164, 18) and (3164, 1)
```

```
The shape of the training records in the dataseti.e., X_test and y_test are (1356, 18) and (1356, 1)
```

#Method 1-Using Sklearn- Without Grid Search CV

The first process being the standard one where in the dataset we apply the Regression model, which is done with help of the package Sklearn and the different modules as required.

For this particular method, we have opted for the XG Boost Regressor and it being there in the Sklearn package

Step 6-

In this step we apply the XG Boost Regressor model on the Train data to check the overall model building process.

We first apply the whole model on the train dataset and then test the model on the testing dataset.

We haven't used the scaling technique as it is not required in the XG Boost Regressor

Actual vs Predicted Values- XG Boost Regressor

	Real Values	Predicted Values for test
0	8.646290	8.448859
1	8.788441	8.982213
2	7.796058	7.954561
3	8.409831	8.473565
4	7.789869	7.763090
...	...	...
1351	8.551788	8.549588
1352	7.774436	7.966436
1353	7.705713	8.130435
1354	7.818430	7.890074
1355	8.425516	8.446533

1356 rows × 2 columns

Metrices in XG Boost Regressor-Table

Metrices	Train	Test	Comments
MSE	**0.22%**	**2.1%**	Here, both the MSE values for Train and Test datasets are very low, which signifies that the Model is better for prediction, as due to lower value, data points are dispersed closely around its central moment
RMSE	**4.7%**	**14.49%**	Here, both the RMSE values for Train and Test datasets are low to moderate, which signifies that the Model is still good for prediction, as due to lower value, spread of the residuals is not too high. There has been a drastic difference between the train and test values
MAPE	**0.41%**	**1.36%**	Here, both the MAPE values for Train and Test datasets are quite low, which signifies that the Model is good for prediction, as due to lower value**,** significantly the accuracy of a forecast system increases
MAE	**3.44%**	**11.21%**	Here, both the MAE values for Train and Test datasets are moderately low, which signifies that the Model is good for prediction, as due to lower value**,** significantly the accuracy of the predictive model increases. There has been a drastic difference between the train and test values
R^2 Score	**98.04%**	**81.36%**	Here, both the MAE values for Train and Test datasets are quite high, which signifies that the Model is overall good for prediction, as due to higher value**,** significantly the independent variables are being able to explain the variance for the target variable

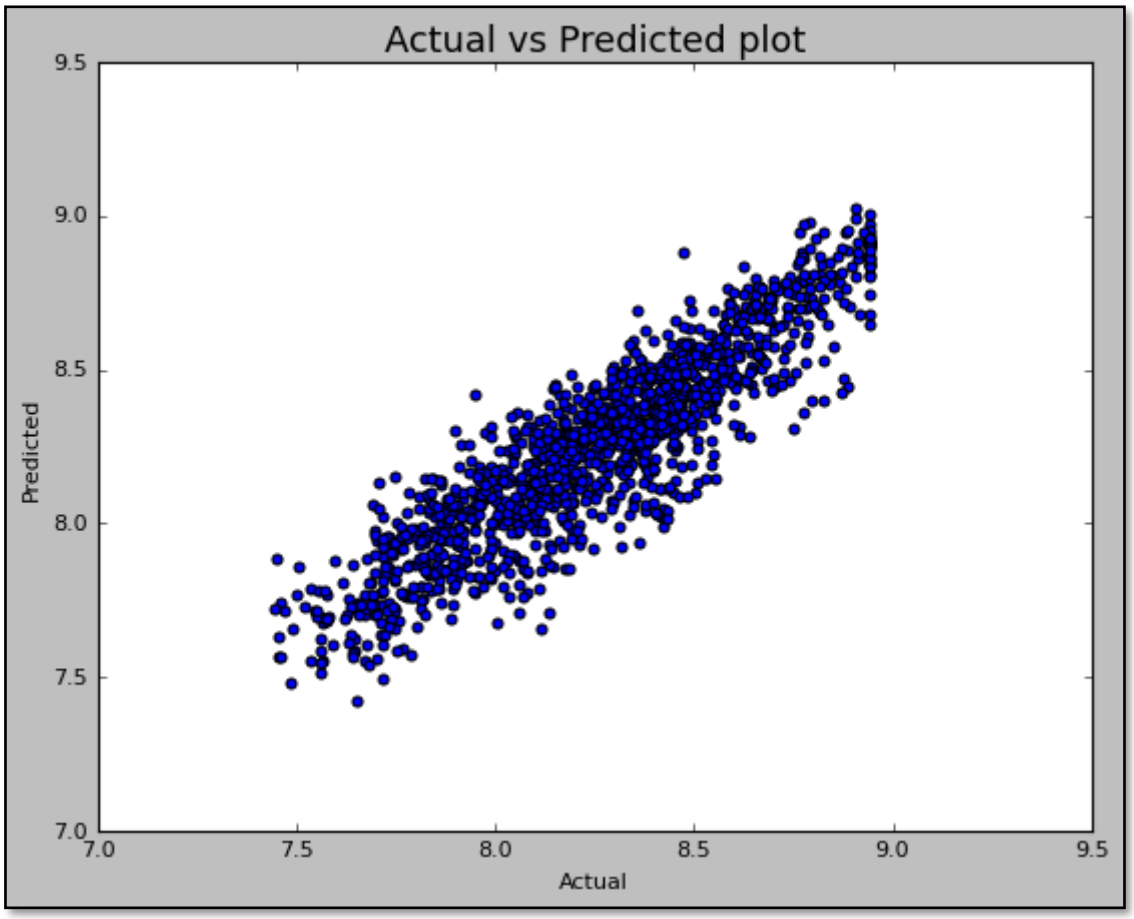
Output in terms of the Actual vs Predicted price plot generated

The output from the qq plot has shown that the predicted and actual price values almost follow a curve line or an area in general, with low-moderate disparity or variance to be precise.

But to presence of multicollinearity, the residual errors are moderately low, causing the dispersion of the data, as observed from the graph.

Also, it shows a more or less linear trend for the residuals

Scatter plot of Actual vs Predicted price- XG Boost Regressor



Interpretation from the models

Considering all the required metrices for comparing the model performances, we can come to the conclusion that the three top most performing models are-

- **Random Forest Regressor**
- **Gradient Boosting Regressor**
- **XG Boost Regressor**

Metrices	Random Forest Regressor		Gradient Boost Regressor		XG Boost Regressor	
	Train	**Test**	**Train**	**Test**	**Train**	**Test**
MSE	0.24%	1.79%	1.52%	1.88%	0.22%	2.10%
RMSE	4.93%	13.40%	12.35%	13.72%	4.70%	14.49%
MAPE	0.46%	1.25%	1.18%	1.32%	0.41%	1.36%
MAE	3.77%	10.30%	9.73%	10.83%	3.44%	11.21%
R2 Score	97.84%	84.04%	86.52%	83.28%	98.04%	81.36%

The residuals or errors are minimum while we follow the above models, which is why is it suggested to use such models while predicting Agent Bonus. Although, we have observed overfitting tendencies in all of them, which is why we will be looking for hyper tuning process in the upcoming stage.

****

4b. Efforts to improve Model Performance

Tuning is usually a trial-and-error process by which you change some hyperparameters (for example, the number of trees in a tree-based algorithm or the value of alpha in a linear algorithm), run the algorithm on the data again, then compare its performance on your validation set in order to determine which set of hyperparameters results in the most accurate model.

It is the process of maximizing a model's performance without overfitting or creating too high of a variance. Hyperparameters can be thought of as the “dials” or “knobs” of a machine learning model.

All machine learning algorithms have a “default” set of hyperparameters, which Machine Learning Mastery defines as “a configuration that is external to the model and whose value cannot be estimated from data.” Different algorithms consist of different hyperparameters. For example, regularized regression models have coefficients penalties, decision trees have a set number of branches, and neural networks have a set number of layers. When building models, analysts and data scientists choose the default configuration of these hyperparameters after running the model on several datasets.

While the generic set of hyperparameters for each algorithm provides a starting point for analysis and will generally result in a well-performing model, it may not have the optimal configurations for your particular dataset and business problem. In order to find the best hyperparameters for your data, you need to tune them.

- ***Why is Model Tuning Important?***

Model tuning allows you to customize your models so they generate the most accurate outcomes and give you highly valuable insights into your data, enabling you to make the most effective business decisions.

Out of several approaches for tuning the hyperparameter, we choose **Grid Search CV** for solving this problem

Grid Search CV

Grid search is a basic method for hyperparameter tuning. It performs an exhaustive search on the hyperparameter set specified by users. This approach is the most straightforward leading to the most accurate predictions. Using this tuning method, users can find the optimal combination. Grid search is applicable for several hyper-parameters, however, with limited search space.

- Exhaustive search over specified parameter values for an estimator.
- Important members are fit, predict.
- GridSearchCV implements a “fit” and a “score” method.

GridSearchCV is a function that comes in Scikit-learn’s(or SK-learn) model_selection package.So an important point here to note is that we need to have the Scikit learn library installed on the computer. This function helps to loop through predefined hyperparameters and fit your estimator (model) on your training set. So, in the end, we can select the best parameters from the listed hyperparameters.

1.estimator: Pass the model instance for which you want to check the hyperparameters.
2.params_grid: the dictionary object that holds the hyperparameters you want to try
3.scoring: evaluation metric that you want to use, you can simply pass a valid string/ object of evaluation metric
4.cv: number of cross-validation you have to try for each selected set of hyperparameters
5.verbose: you can set it to 1 to get the detailed print out while you fit the data to GridSearchCV
6.n_jobs: number of processes you wish to run in parallel for this task if it -1 it will use all available processors.

We haven’t shown the details of the hyper parameter tuning from Ridge and Lasso regression, as the values coming were not at par with other models

DECISION TREE REGRESSOR

#Method 2-Using Sklearn- Grid Search CV

We will only discuss in details about the model building process after using Grid Search CV and the respective outputs. (Rest of the steps are already being explained in the normal model building for each of the processes)

Step 1 -

In this step we apply the GRID Search CV on the whole model in order to optimise the same

GRID Search CV – best model hyperparameters obtained

Use the best parameter to get better model performance

Best: -0.036441 using {'max_depth': 5,'max_features': 'auto', 'max_leaf_nodes': 10,'min_samples_leaf': 1, 'min_weight_fraction_leaf': 0.1,'splitter': 'best'}

Using the above mentioned GridSearchCV package, we have identified the best parameters and an Optimised Logistic Regression model is being built after doing few iterations with the values we have received in each step. But even with this the model couldn't generate better accuracy, precision and recall. We will discuss about the same in the next questions

Note – Kindly refer the code file for the steps involved in the CART formation.

Step 2-

In this step we apply the Decision Tree Regressor model on the Train data to check the overall model building process.

We first apply the whole model on the train dataset and then test the model on the testing dataset.

Actual vs Predicted Values- Decision Tree Regressor-After using Gridsearch

	Real Values	Predicted Values for test
0	8.646290	8.484187
1	8.788441	8.698462
2	7.796058	8.146907
3	8.409831	8.698462
4	7.789869	7.849891
...	...	...
1351	8.551788	8.146907
1352	7.774436	8.015655
1353	7.705713	8.146907
1354	7.818430	7.849891
1355	8.425516	8.484187

1356 rows × 2 columns

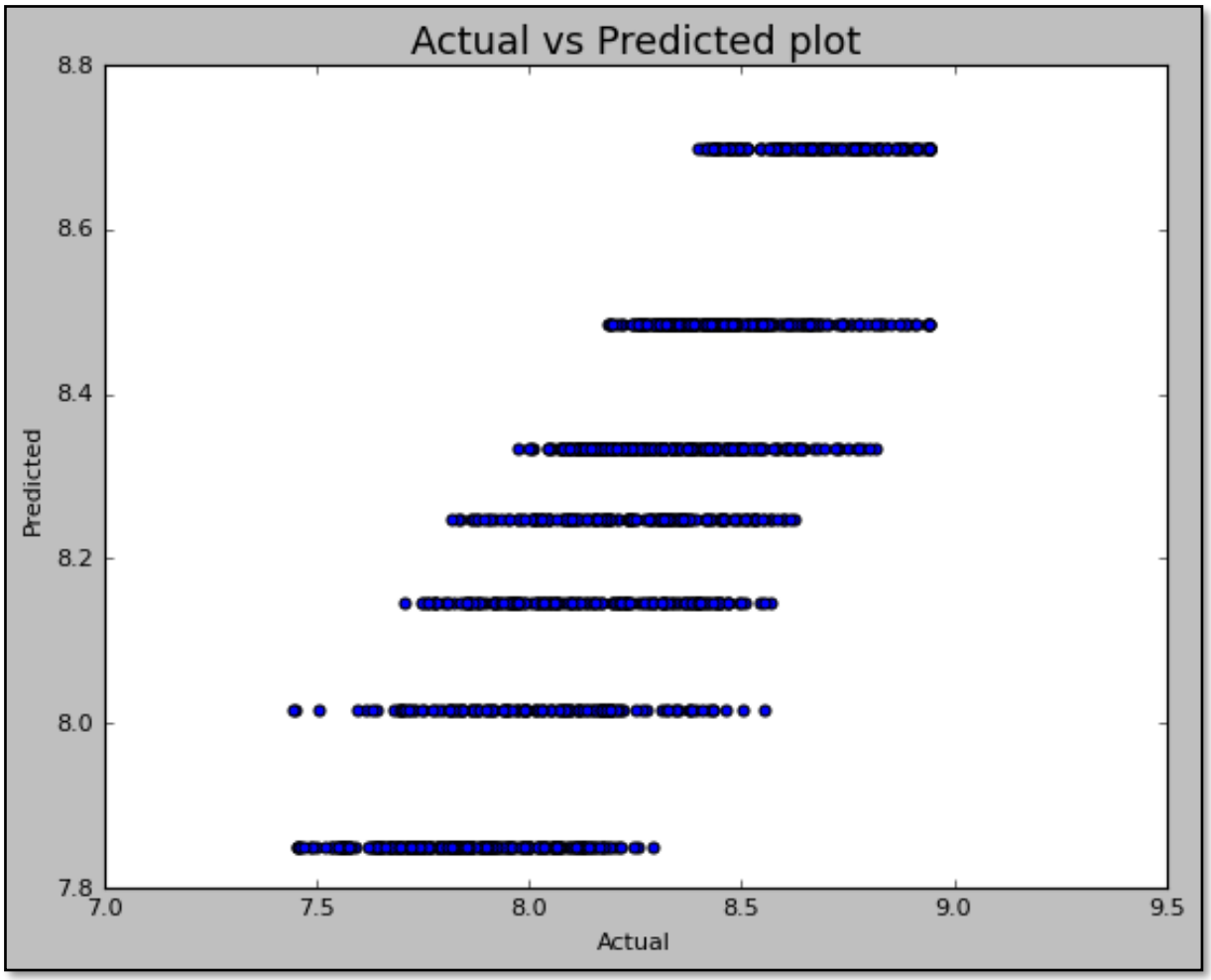
Metrices in Decision Tree Regressor (Using Grid Search)-Table

Metrices	Train	Test	Comments
MSE	**3.43%**	**3.65%**	Here, both the MSE values for Train and Test datasets are quite low, which signifies that the Model is better for prediction, as due to lower value, data points are dispersed closely around its central moment
RMSE	**18.52%**	**19.12%**	Here, both the RMSE values for Train and Test datasets are quite low, which signifies that the Model is better for prediction, as due to lower value, spread of the residuals is low
MAPE	**18.50%**	**19.19%**	Here, both the MAPE values for Train and Test datasets are quite low, which signifies that the Model is good for prediction, as due to lower **value,** significantly the accuracy of a forecast system increases
MAE	**15.22%**	**15.79%**	Here, both the MAE values for Train and Test datasets are quite low, which signifies that the Model is good for prediction, as due to lower **value,** significantly the accuracy of the predictive model increases
R^2 Score	**69.72%**	**67.56%**	Here, both the MAE values for Train and Test datasets are quite low, which signifies that the Model is not too for prediction, as due to lower value, significantly the independent variables are not being able to explain the variance for the target variable

Output in terms of the Actual vs Predicted price plot generated

The output from the qq plot has shown that the predicted and actual price values almost are difficult to interpret from the graph

Scatter plot of Actual vs Predicted price- Decision Tree Regressor - Using Grid Search



RANDOM FOREST REGRESSOR

#Method 2-Using Sklearn- Grid Search CV

We will only discuss in details about the model building process after using Grid Search CV and the respective outputs. (Rest of the steps are already being explained in the normal model building for each of the processes)

Step 1 -

In this step we apply the GRID Search CV on the whole model in order to optimise the same

GRID Search CV – best model hyperparameters obtained

Use the best parameter to get better model performance

Best: 0.804566 using {'bootstrap': True, 'max_depth': 80, 'max_features': 3, 'min_samples_leaf': 3, 'min_samples_split': 8, 'n_estimators': 200}

Using the above mentioned GridSearchCV package, we have identified the best parameters and an Optimised Logistic Regression model is being built after doing few iterations with the values we have received in each step. But even with this the model couldn't generate better accuracy, precision and recall. We will discuss about the same in the next questions

Note – Kindly refer the code file for the steps involved in the CART formation.

Step 2-

In this step we apply the Random Forest Regressor model on the Train data to check the overall model building process.

We first apply the whole model on the train dataset and then test the model on the testing dataset.

Actual vs Predicted Values-Random Forest Regressor-After using Gridsearch

	Real Values	Predicted Values for test
0	8.646290	8.433834
1	8.788441	8.727207
2	7.796058	8.030337
3	8.409831	8.428362
4	7.789869	7.772697
...	...	...
1351	8.551788	8.511014
1352	7.774436	8.033386
1353	7.705713	7.955269
1354	7.818430	7.860684
1355	8.425516	8.510451

1356 rows × 2 columns

Metrices in Random Forest Regressor (Using Grid Search)-Table

Metrices	Train	Test	Comments
MSE	**1.10%**	**2.13%**	Here, both the MSE values for Train and Test datasets are quite low, which signifies that the Model is better for prediction, as due to lower value, data points are dispersed closely around its central moment
RMSE	**10.05%**	**14.62%**	Here, both the RMSE values for Train and Test datasets are quite low, which signifies that the Model is better for prediction, as due to lower value, spread of the residuals is low
MAPE	**0.98%**	**1.43%**	Here, both the MAPE values for Train and Test datasets are quite low, which signifies that the Model is good for prediction, as due to lower value**,** significantly the accuracy of a forecast system increases
MAE	**3.77%**	**10.30%**	Here, both the MAE values for Train and Test datasets are quite low, which signifies that the Model is good for prediction, as due to lower value**,** significantly the accuracy of the predictive model increases. But there is a significant increase in the Test data value as compared to the Train dataset
R^2 Score	**91.08%**	**81.03%**	Here, both the MAE values for Train and Test datasets are quite high, which signifies that the Model is good for prediction, as due to higher value**,** significantly the independent variables are being able to explain the variance for the target variable

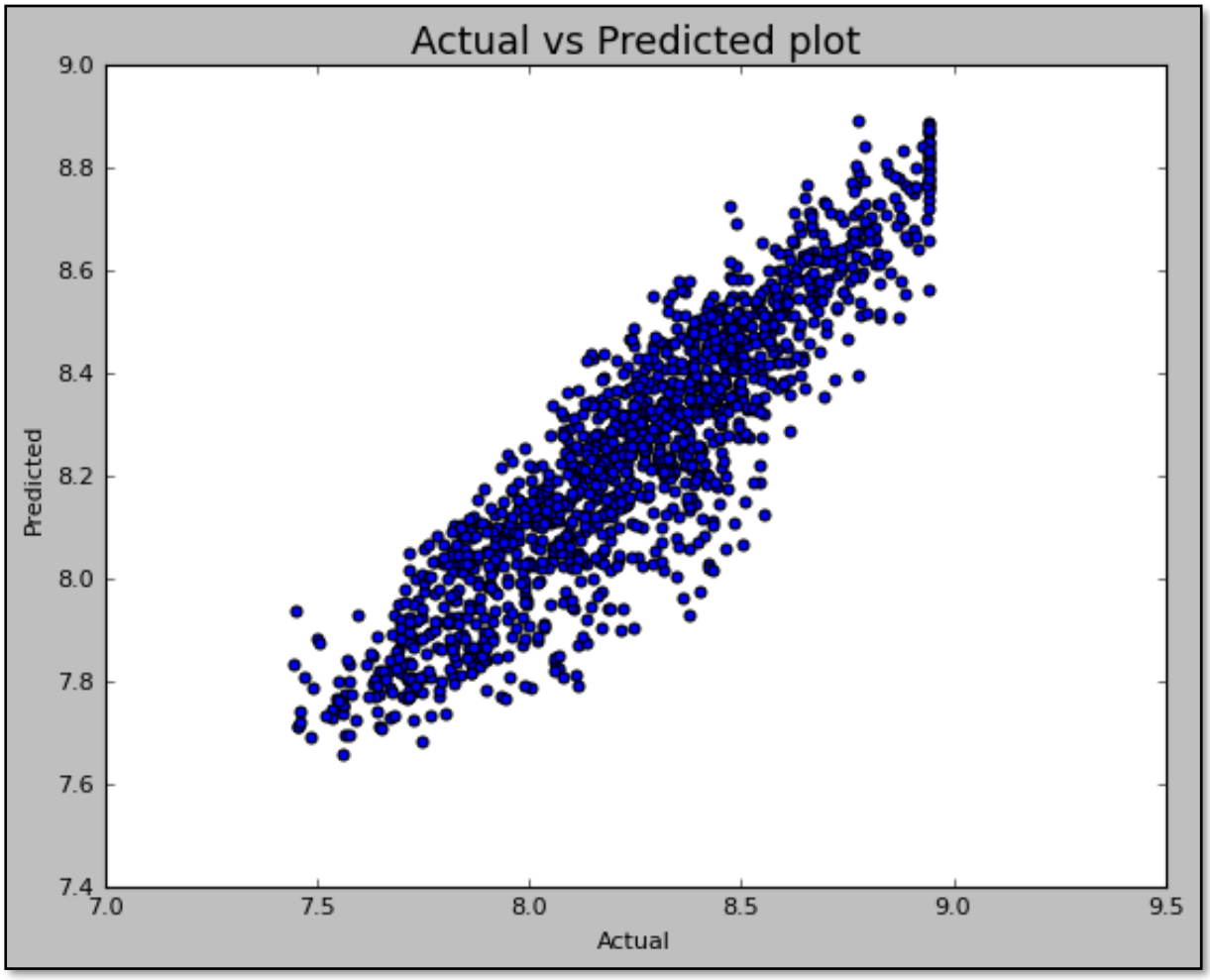
Output in terms of the Actual vs Predicted price plot generated

The output from the qq plot has shown that the predicted and actual price values almost follow a curve line or an area in general, with low-moderate disparity or variance to be precise.

But to presence of multicollinearity, the residual errors are high, causing the dispersion of the data, as observed from the graph.

Also, it shows a more or less linear trend for the residuals, with tapering ends

Scatter plot of Actual vs Predicted price- Random Forest Regressor - Using Grid Search



K-NEAREST NEIGHBOUR

#Method 2-Using Sklearn- Grid Search CV

We will only discuss in details about the model building process after using Grid Search CV and the respective outputs. (Rest of the steps are already being explained in the normal model building for each of the processes)

Step 1 -

In this step we apply the GRID Search CV on the whole model in order to optimise the same

GRID Search CV – best model hyperparameters obtained

Use the best parameter to get better model performance

Best: 0.71712 using {'algorithm': 'kd_tree','leaf_size': 2,'metric': 'cityblock','n_neighbors': 6,'weights': 'uniform'}

Using the above mentioned GridSearchCV package, we have identified the best parameters and an Optimised Logistic Regression model is being built after doing few iterations with the values we have received in each step. But even with this the model couldn't generate better accuracy, precision and recall. We will discuss about the same in the next questions

Note – Kindly refer the code file for the steps involved in the CART formation.

Step 2-

In this step we apply the KNN model on the Train data to check the overall model building process.

We first apply the whole model on the train dataset and then test the model on the testing dataset.

Actual vs Predicted Values-KNN-After using Gridsearch

	Real Values	Predicted Values for test
0	8.646290	8.325533
1	8.788441	8.677166
2	7.796058	8.095095
3	8.409831	8.243510
4	7.789869	8.063991
...	...	...
1351	8.551788	8.459911
1352	7.774436	8.070768
1353	7.705713	8.052126
1354	7.818430	8.083420
1355	8.425516	8.551283

1356 rows × 2 columns

Metrices in KNN Regressor (Using Grid Search)-Table

Metrices	Train	Test	Comments
MSE	**5.00%**	**6.95%**	Here, both the MSE values for Train and Test datasets are low, which signifies that the Model is better for prediction, as due to lower value, data points are dispersed closely around its central moment
RMSE	**22.37%**	**26.36%**	Here, both the RMSE values for Train and Test datasets are low to moderate, which signifies that the Model is still good for prediction, as due to lower value, spread of the residuals is not too high
MAPE	**2.25%**	**2.68%**	Here, both the MAPE values for Train and Test datasets are quite low, which signifies that the Model is good for prediction, as due to lower value**,** significantly the accuracy of a forecast system increases
MAE	**18.53%**	**22.10%**	Here, both the MAE values for Train and Test datasets are moderately low, which signifies that the Model is good for prediction, as due to lower value**,** significantly the accuracy of the predictive model increases.
R^2 Score	**55.83%**	**38.33%**	Here, both the MAE values for Train and Test datasets are quite low, which signifies that the Model is too bad for prediction, as due to lower value**,** significantly the independent variables are not being able to explain the variance for the target variable

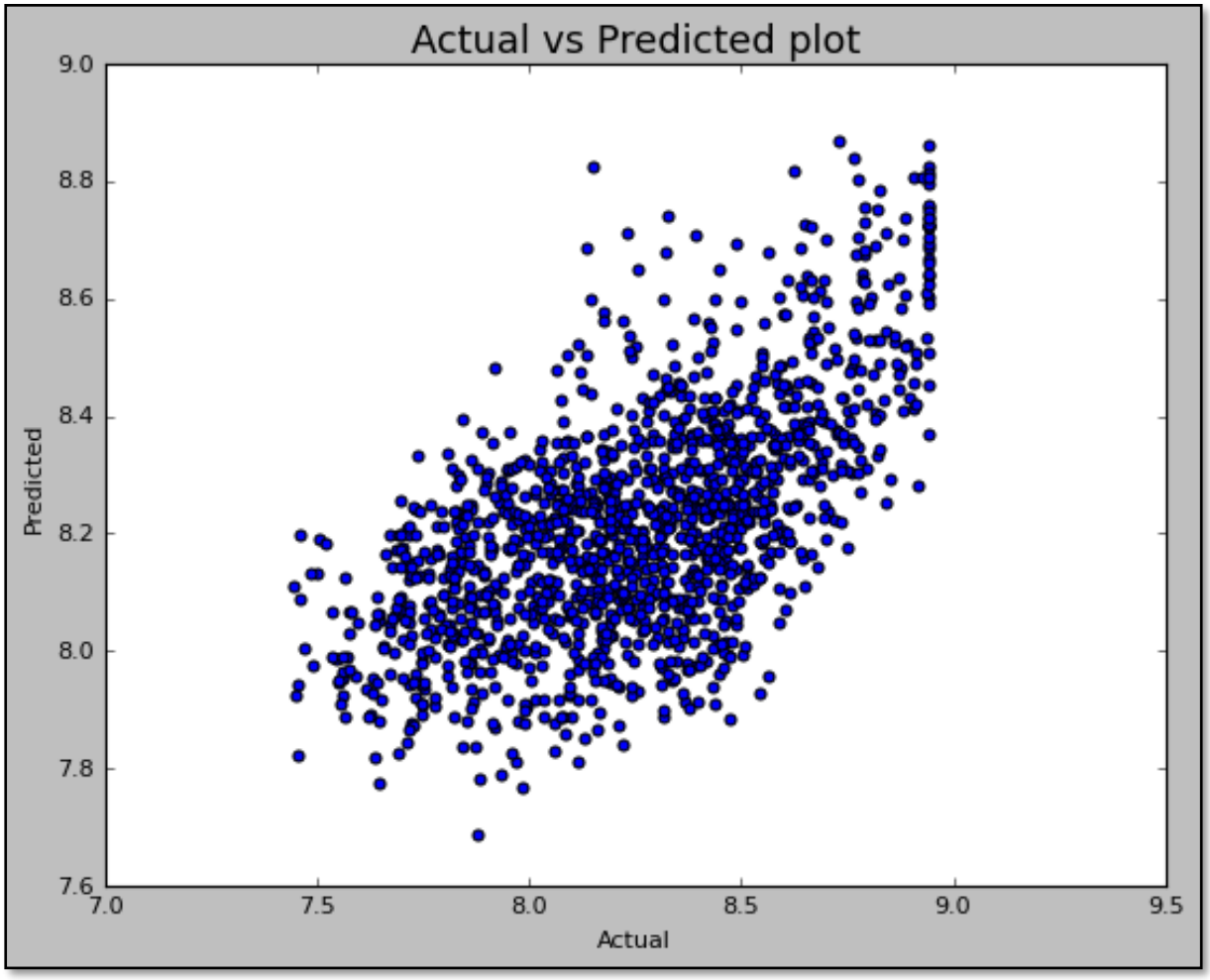
Output in terms of the Actual vs Predicted price plot generated

The output from the qq plot has shown that the predicted and actual price values almost form a clutter due to high disparity

But to presence of multicollinearity, the residual errors are high, causing the dispersion of the data, as observed from the graph.

We are unable to find a trend from this graph

Scatter plot of Actual vs Predicted price-KNN- Using Grid Search



GRADIENT BOOSTING METHOD

#Method 2-Using Sklearn- Grid Search CV

We will only discuss in details about the model building process after using Grid Search CV and the respective outputs. (Rest of the steps are already being explained in the normal model building for each of the processes)

Step 1 -

In this step we apply the GRID Search CV on the whole model in order to optimise the same

GRID Search CV – best model hyperparameters obtained

Use the best parameter to get better model performance

Best: 0.800339 using {"learning_rate': 0.01, 'max_depth': 4, 'n_estimators': 1000, 'subsample': 0.9}

Using the above mentioned GridSearchCV package, we have identified the best parameters and an Optimised Logistic Regression model is being built after doing few iterations with the values we have received in each step. But even with this the model couldn't generate better accuracy, precision and recall. We will discuss about the same in the next questions

Note – Kindly refer the code file for the steps involved in the CART formation.

Step 2-

In this step we apply the Gradient Boosting model on the Train data to check the overall model building process.

We first apply the whole model on the train dataset and then test the model on the testing dataset.

Actual vs Predicted Values- Gradient Boosting Regressor-After using Gridsearch

	Real Values	Predicted Values for test
0	8.646290	8.433834
1	8.788441	8.727207
2	7.796058	8.030337
3	8.409831	8.428362
4	7.789869	7.772697
...	...	...
1351	8.551788	8.511014
1352	7.774436	8.033386
1353	7.705713	7.955269
1354	7.818430	7.860684
1355	8.425516	8.510451

1356 rows × 2 columns

Metrices in Gradient Boosting Regressor (Using Grid Search)-Table

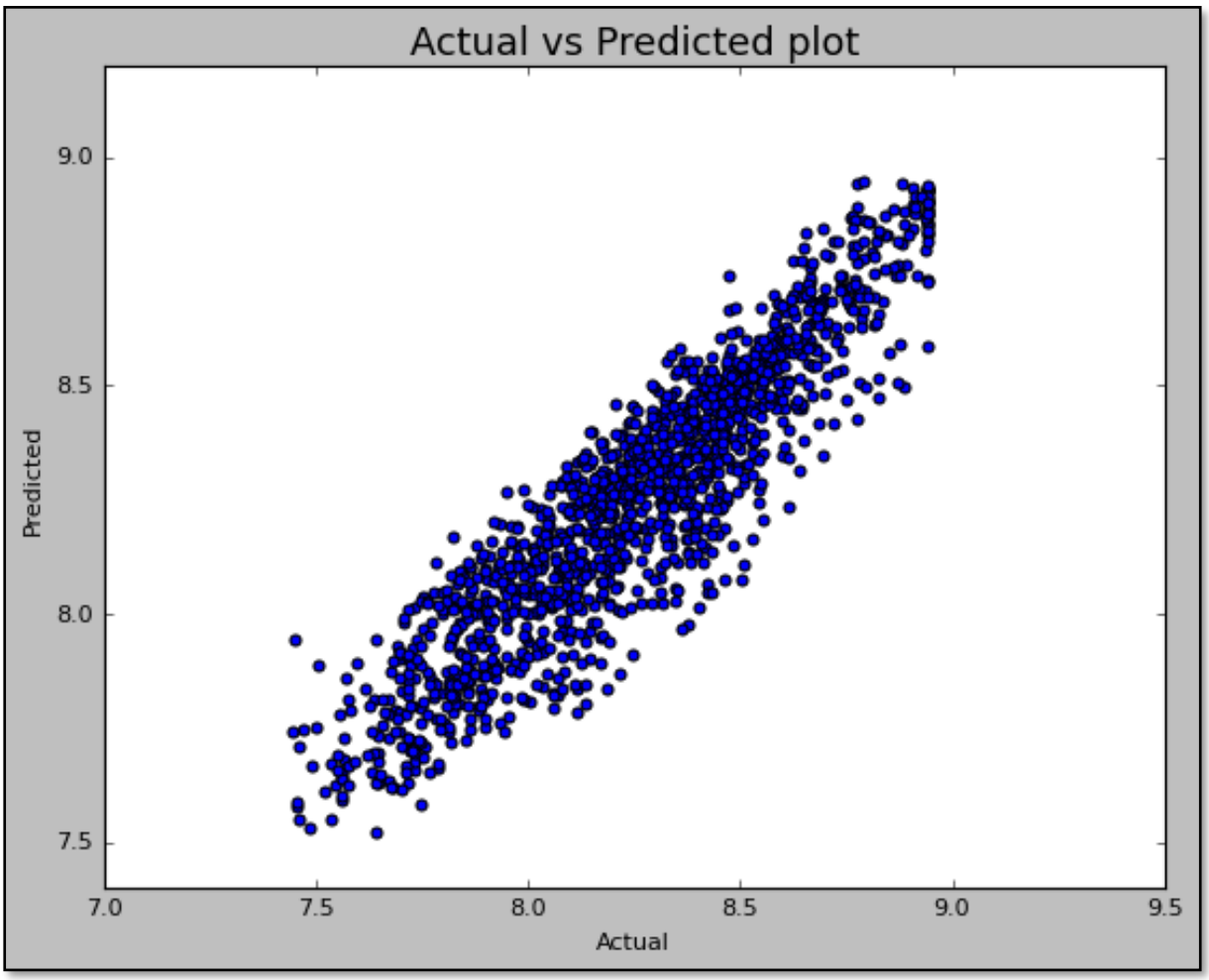
Metrices	Train	Test	Comments
MSE	**1.20%**	**1.79%**	Here, both the MSE values for Train and Test datasets are low, which signifies that the Model is better for prediction, as due to lower value, data points are dispersed closely around its central moment
RMSE	**10.96%**	**13.40%**	Here, both the RMSE values for Train and Test datasets are low to moderate, which signifies that the Model is still good for prediction, as due to lower value, spread of the residuals is not too high
MAPE	**1.18%**	**1.32%**	Here, both the MAPE values for Train and Test datasets are quite low, which signifies that the Model is good for prediction, as due to lower value**,** significantly the accuracy of a forecast system increases
MAE	**8.57%**	**10.47%**	Here, both the MAE values for Train and Test datasets are moderately low, which signifies that the Model is good for prediction, as due to lower value**,** significantly the accuracy of the predictive model increases.
R^2 Score	**89.38%**	**84.07%**	Here, both the MAE values for Train and Test datasets are quite high, which signifies that the Model is overall good for prediction, as due to higher value**,** significantly the independent variables are being able to explain the variance for the target variable

Output in terms of the Actual vs Predicted price plot generated

The output from the qq plot has shown that the predicted and actual price values almost follow a curve line or an area in general, with low-moderate disparity or variance to be precise.

But to presence of multicollinearity, the residual errors are high, causing the dispersion of the data, as observed from the graph.

Scatter plot of Actual vs Predicted price- Gradient Boosting Regressor- Using Grid Search



XG BOOSTING METHOD

#Method 2-Using Sklearn- Grid Search CV

We will only discuss in details about the model building process after using Grid Search CV and the respective outputs. (Rest of the steps are already being explained in the normal model building for each of the processes)

Step 1 -
In this step we apply the GRID Search CV on the whole model in order to optimise the same

GRID Search CV – best model hyperparameters obtained

Use the best parameter to get better model performance

Best: -0.01754 using {'colsample_bytree': 0.7,'learning_rate': 0.01,'max_depth': 10,'n_estimators': 1000}

Using the above mentioned GridSearchCV package, we have identified the best parameters and an Optimised Logistic Regression model is being built after doing few iterations with the values we have received in each step. But even with this the model couldn't generate better accuracy, precision and recall. We will discuss about the same in the next questions

Note – Kindly refer the code file for the steps involved in the CART formation.

Step 2-

In this step we apply the XGBoost model on the Train data to check the overall model building process.
We first apply the whole model on the train dataset and then test the model on the testing dataset.

Actual vs Predicted Values-XG Boosting Regressor-After using Gridsearch

	Real Values	Predicted Values for test
0	8.646290	8.456025
1	8.788441	8.856379
2	7.796058	8.011599
3	8.409831	8.488452
4	7.789869	7.751311
...	...	...
1351	8.551788	8.567135
1352	7.774436	7.966618
1353	7.705713	8.002942
1354	7.818430	7.846949
1355	8.425516	8.518282

1356 rows × 2 columns

Metrices in XG Boosting Regressor (Using Grid Search)-Table

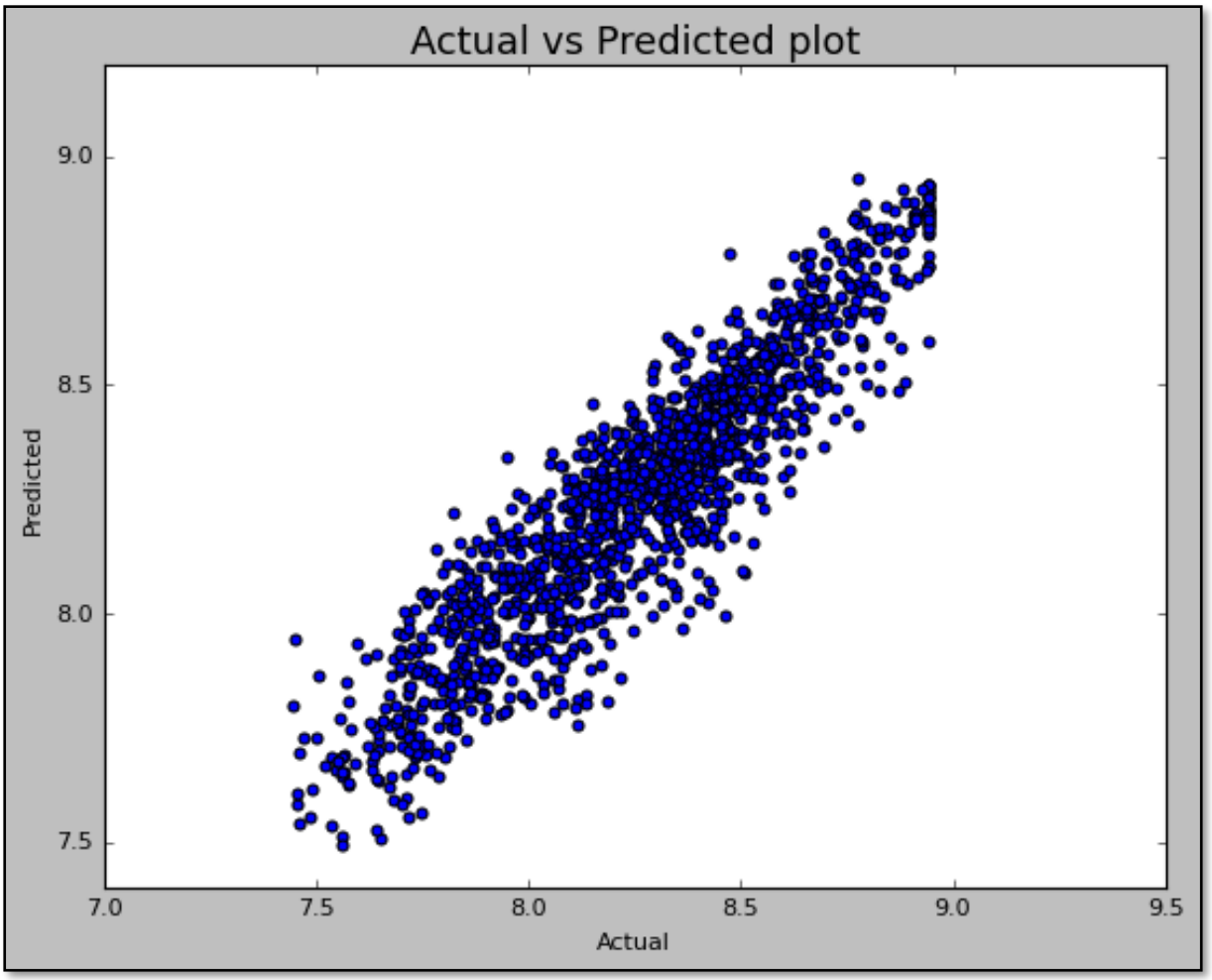
Metrices	Train	Test	Comments
MSE	**0.19%**	**1.82%**	Here, both the MSE values for Train and Test datasets are low, which signifies that the Model is better for prediction, as due to lower value, data points are dispersed closely around its central moment
RMSE	**4.70%**	**14.49%**	Here, both the RMSE values for Train and Test datasets are low to moderate, which signifies that the Model is still good for prediction, as due to lower value, spread of the residuals is not too high
MAPE	**0.41%**	**1.36%**	Here, both the MAPE values for Train and Test datasets are quite low, which signifies that the Model is good for prediction, as due to lower value**,** significantly the accuracy of a forecast system increases
MAE	**3.44%**	**11.21%**	Here, both the MAE values for Train and Test datasets are moderately low, which signifies that the Model is good for prediction, as due to lower value**,** significantly the accuracy of the predictive model increases.
R^2 Score	**98.04%**	**81.36%**	Here, both the MAE values for Train and Test datasets are quite high, which signifies that the Model is overall good for prediction, as due to higher value**,** significantly the independent variables are being able to explain the variance for the target variable

Output in terms of the Actual vs Predicted price plot generated

The output from the qq plot has shown that the predicted and actual price values almost follow a curve line or an area in general, with low-moderate disparity or variance to be precise.

But to presence of multicollinearity, the residual errors are high, causing the dispersion of the data, as observed from the graph.

Scatter plot of Actual vs Predicted price-XG Boosting Regressor- Using Grid Search



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

Model Comparison

Considering all the required metrices for comparing the model performances, we can come to the conclusion that the three top most performing models are-

- **Gradient Boosting Regressor**
- **Random Forest Regressor**
- **XG Boost Regressor**

Metrices	Random Forest Regressor		Gradient Boost Regressor		XG Boost Regressor	
	Train	**Test**	**Train**	**Test**	**Train**	**Test**
MSE	1.10%	2.13%	1.20%	1.79%	0.19%	1.82%
RMSE	10.05%	14.62%	10.96%	13.40%	4.70%	14.49%
MAPE	0.98%	1.43%	1.18%	1.32%	0.41%	1.36%
MAE	3.77%	10.30%	8.57%	10.47%	3.44%	11.21%
R2 Score	91.08%	81.03%	89.38%	84.07%	98.04%	81.36%

As per all the models being considered, **Gradient Boosting Regressor Algorithm** is being considered as the best model on the basis of all the parameters.

****

5. Model Validation

How was the model validated? Just accuracy, or anything else too?

As already stated, before the model building process, there are several metrices when it comes to Regression problems. Hence, we have considered such metrices and compared them w.r.t. each model in order to understand, which one of the models works best in terms of predicting the Bonus of the Agents.
The details of each of the metrices have already been discussed (kindly refer to pg no. 34-36)
As per all the models being considered, **Gradient Boosting Regressor Algorithm** is being considered as the best model on the basis of all the parameters.

Metrices		MSE	RMSE	MAPE	MAE	R2 Score
Random Forest Regressor	**Train**	1.10%	10.05%	0.98%	3.77%	91.08%
	Test	2.13%	14.62%	1.43%	10.30%	81.03%
Gradient Boost Regressor	**Train**	1.20%	10.96%	1.18%	8.57%	89.38%
	Test	1.79%	13.40%	1.32%	10.47%	84.07%
XG Boost Regressor	**Train**	0.19%	4.70%	0.41%	3.44%	98.04%
	Test	1.82%	14.49%	1.36%	11.21%	81.36%
Linear Regressor	**Train**	2.67%	16.35%	1.60%	13.15%	76.39%
	Test	2.77%	16.65%	1.64%	13.49%	75.40%
Ridge Regressor	**Train**	2.67%	16.35%	1.60%	13.15%	76.39%
	Test	2.77%	16.65%	1.64%	13.49%	75.40%
Lasso Regressor	**Train**	4.55%	21.33%	2.05%	16.19%	59.85%
	Test	4.65%	21.57%	2.09%	17.15%	58.71%
Decision Tree Regressor	**Train**	3.43%	18.52%	18.50%	15.22%	69.72%
	Test	3.65%	19.12%	19.19%	15.79%	67.56%
KNN Regressor	**Train**	5.00%	22.37%	2.25%	18.53%	55.83%
	Test	6.95%	26.36%	2.68%	22.10%	38.33%

****

6. Final interpretation / recommendation

Business insights from EDA

As per the business insights being concerned-

- On an average, bonus per agent is approx 4000 units
- AgentBonus increases with increase in SumAssured, which in general means, that if a higher premium product can e pitched to a customer by an Agent, he is likely to receive higher Bonus
- Monthly Income has a direct relationship with Agent Bonus, which means that with increase in Income, Bonus increases
- Even the Customer tenure has an impact with the Agent Bonus, meaning, longer the Tenure of the Customer, more Bonus will be allocated to the Agents
- LastMonthCalls, which are basically calls by the Agents to cross sell the products to the customers, doesn't contribute much to the Bonus received
- Even, we have seen that the Customer Care Score, which being the highest, contribute to Higher bonuses. Thus, Agent performance is very much vital and how an Agent manages the After Sales service and serves the Customers, and this plays an important role to the Bonus received.
- The Designation of the customers also plays an important role as on the basis of Higher Income range for Customers (ex- Vice President, Assistant Vice President, etc), they tend to buy higher premium policies, which in turn influence the Bonus of the Agents
- The Agents should be trained to sell the products ethically, and focus not just on selling but rather on the problems that they can provide a solution to by suggesting the right kind of products that cater the needs of the customers. Even the After Sales service is important to keep the customers.
- Depending on the overall demographics, the Agents should study the market first, based on the type of customers looking for certain kinds of Insurance

Business insights from Modelling

As per the Business implications are being concerned, it is important to understand the importance of each feature and the purpose it serves.
Since the very start we tried several models to predict the right Bonus amount for the Agents and out of that only the above three could provide significant results

As per the business insights being concerned-

The most important features affecting the Agent Bonus are-

- Sum Assured,
- Age
- CustTenure
- Monthly Income
- Designation

The companies can use such models to reward their Agents or train them as required
We have already mentioned the below points before while doing EDA as well

- With the help of modelling techniques, the companies can figure out the new parameters that can be used in order to predict the Agent Bonus
- **Mastering technological change -** By replacing non-transparent, outdated and historically grown IT systems, maintenance costs and the complexity of the IT infrastructure can be reduced. Modern, integrated solutions increase the flexibility for the implementation of business requirements. Technologies can help in a bigger way to shape the profitability of the Business, which in turn will help the Agents with their Bonus
- **Staying ahead of the competition -** The integration of new digital processes and the redesign of compensation components helps insurers to compete against disruptive insurTechs. The "grazing" of portfolio commissions and ultimately the takeover of the entire customer relationship can be counteracted. Like it has been pointed out, if the Agents can make greater use of the Digital Platforms and the accessibility and visibility that it offers, it can help them to educate more customers and generate lead accordingly benefiting them and the business.
- **Mapping changed customer behaviour** - The distribution channels and sales management will be adapted to the changed market conditions. In addition, consistent and action-based sales remuneration ensures the implementation of customer-centric sales strategies (such as omni-channel strategies). This is one of the important factors that can help the Agents to focus on customers and help them reap benefits by understanding the behaviour change and pitch the policies accordingly

****

References

- *https://www.investopedia.com/articles/fa-profession/093016/5-ways-advisors-can-increase-sales-life-insurance.asp*
- *https://www.spcforexcel.com/knowledge/basic-statistics/are-skewness-and-kurtosis-useful-statistics#:~:text=If%20the%20skewness%20is%20between,the%20data%20are%20highly%20skewed*
- *https://www.seldon.io/supervised-vs-unsupervised-learning-explained#:~:text=Supervised%20vs%20unsupervised%20learning%20compared&text=Supervised%20machine%20learning%20relies%20on,labelled%20input%20and%20output%20data.*
- *https://towardsdatascience.com/all-about-categorical-variable-encoding-305f3361fd02*
- *https://www.analyticsvidhya.com/blog/2021/05/feature-scaling-techniques-in-python-a-complete-guide/*
- *https://stackabuse.com/feature-scaling-data-with-scikit-learn-for-machine-learning-in-python/*
- *https://datascience.stackexchange.com/questions/9443/when-to-use-one-hot-encoding-vs-labelencoder-vs-dictvectorizor#:~:text=One%2DHot%2DEncoding%20has%20the,with%20the%20curse%20of%20dimensionality.*
- *https://www.geeksforgeeks.org/data-pre-processing-wit-sklearn-using-standard-and-minmax-scaler/#:~:text=MinMax%20Scaler%20shrinks%20the%20data,shape%20of%20the%20original%20distribution.*
- *https://towardsdatascience.com/everything-you-need-to-know-about-min-max-normalization-in-python-b79592732b79*
- *https://vitalflux.com/minmaxscaler-standardscaler-python-examples/*
- *https://www.statisticshowto.com/probability-and-statistics/regression-analysis/rmse-root-mean-square-error/*
- *https://help.sap.com/docs/SAP_PREDICTIVE_ANALYTICS/41d1a6d4e7574e32b815f1cc87c00f42/5e5198fd4afe4ae5b48fefe0d3161810.html?locale=en-US*
- *https://www.statisticshowto.com/probability-and-statistics/statistics-definitions/mean-squared-error/*
- *https://towardsdatascience.com/mad-over-mape-a86a8d831447#:~:text=MAPE%3A%20Mean%20Absolute%20Percentage%20Error,actual%20data%20at%20time%20i.*
- *https://www.statisticshowto.com/mean-absolute-percentage-error-mape/*
- *https://medium.com/human-in-a-machine-world/mae-and-rmse-which-metric-is-better-e60ac3bde13d*
- *https://www.saedsayad.com/decision_tree_reg.htm#:~:text=Decision%20tree%20builds%20regression%20or,decision%20nodes%20and%20leaf%20nodes.*

- *https://www.upgrad.com/blog/pros-and-cons-of-decision-tree-regression-in-machine-learning/*
- *https://www.datarobot.com/wiki/tuning/*
- *https://analyticsindiamag.com/top-8-approaches-for-tuning-hyperparameters-of-machine-learning-models/*
- *https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html*
- *https://www.mygreatlearning.com/blog/gridsearchcv/*
- *https://www.geeksforgeeks.org/xgboost-for-regression/*
- *https://blog.paperspace.com/implementing-gradient-boosting-regression-python/#:~:text=Gradient%20boosting%20Regression%20calculates%20the,maps%20features%20to%20that%20residual.*
- *https://discuss.boardinfinity.com/t/gradient-boosting-advantages-and-disadvantages/12577*
- *https://towardsdatascience.com/hyperparameter-tuning-the-random-forest-in-python-using-scikit-learn-28d2aa77dd74*
- *https://www.projectpro.io/recipes/find-optimal-parameters-using-gridsearchcv-for-regression*
- *https://towardsdatascience.com/xgboost-fine-tune-and-optimize-your-model-23d996fab663*
- *https://towardsdatascience.com/ridge-regression-for-better-usage-2f19b3a202db*
- *https://www.mygreatlearning.com/blog/what-is-ridge-regression/*
- *https://www.analyticsvidhya.com/blog/2017/06/a-comprehensive-guide-for-linear-ridge-and-lasso-regression/*
- *https://www.analyticsvidhya.com/blog/2021/11/study-of-regularization-techniques-of-linear-model-and-its-roles/*
- *https://www.engati.com/glossary/ridge-regression*
-
-

****

THANK YOU