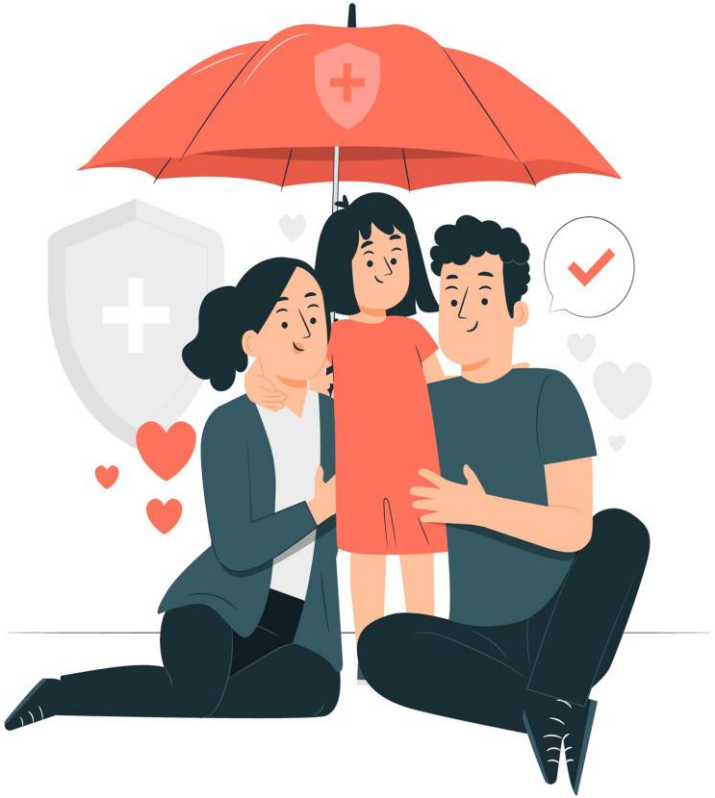


Life Insurance Bonus Prediction

A final Project Presentation for 5302- Principles of Data Science

By

Team 10

Ramya Sai Donkeshwaram

Yuvateja Pothagunta

Problem Statement

To predict the agent bonus based on the Life Insurance policy and also based on type of customers, the agents are bringing in.

So, to reward the valuable agents accordingly and create more engagement programs, and to design new activities accordingly.

Data

Dataset Shape

Our original dataset has,

4520 entries with no duplicates

And 20 columns of categorical and numerical types.

With size 4520 x 20 = 29,000.

Within that, a few missing values ✗

Target column: **AgentBonus**

Data Dictionary

S.No.	Variable	Discription
1	CustID	Unique customer ID
2	AgentBonus	Bonus amount given to each agents in last month
3	Age	Age of customer
4	CustTenure	Tenure of customer in organization
5	Channel	Channel through which acquisition of customer is done
6	Occupation	Occupation of customer
7	EducationField	Field of education of customer
8	Gender	Gender of customer
9	ExistingProdType	Existing product type of customer
10	Designation	Designation of customer in their organization
11	NumberOfPolicy	Total number of existing policy of a customer
12	MaritalStatus	Marital status of customer
13	MonthlyIncome	Gross monthly income of customer
14	Complaint	Indicator of complaint registered in last one month by customer
15	ExistingPolicyTenure	Max tenure in all existing policies of customer
16	SumAssured	Max of sum assured in all existing policies of customer
17	Zone	Customer belongs to which zone in India. Like East, West, North and South
18	PaymentMethod	Frequency of payment selected by customer like Monthly, quarterly, half yearly and yearly
19	LastMonthCalls	Total calls attempted by company to a customer for cross sell
20	CustCareScore	Customer satisfaction score given by customer in previous service call

Corrections from DataFrame information

When we read the data, few categorical columns were wrongly identified as numerical type by Pandas, so we used the below pandas function to correct them

pandas.DataFrame.astype(dtype)

Column	Read as	Which are in fact
CustID	int64	object
ExistingProdType	int64	object
Complaint	int64	object

Data Cleaning - Strings

There are few duplicate values for few categorical columns, due to spelling mistakes or simply because of usage of different terms, so, to maintain the integrity of data, we had to clean them.

We used below function to correct them -


```
df['Column_Name'] =
df['Column_Name'].str.replace('old string','New String')
```

Data Cleaning 1/3

Column	Before Cleaning	After Cleaning
Gender	• Male • Fe male • Female	• Male • Female
Occupation	• Salaried • Small Business • Large Business • Laarge Business • Free Lancer	• Salaried • Small Business • Large Business • Free Lancer

Data Cleaning 2/3

Column	Before Cleaning	After Cleaning
Marital Status	• Married • Single • Divorced • Unmarried	• Married • Single • Divorced
EducationField	• Graduate • Under Graduate • Diploma • Engineer • Post Graduate • UG • MBA	• Graduate • Under Graduate • Diploma • Engineer • Post Graduate • MBA

Data Cleaning 3/3

Column	Before Cleaning	After Cleaning
Designation	• Manager • Executive • Senior Manager • AVP • VP • Exe	• Manager • Executive • Senior Manager • AVP • VP

Outliers

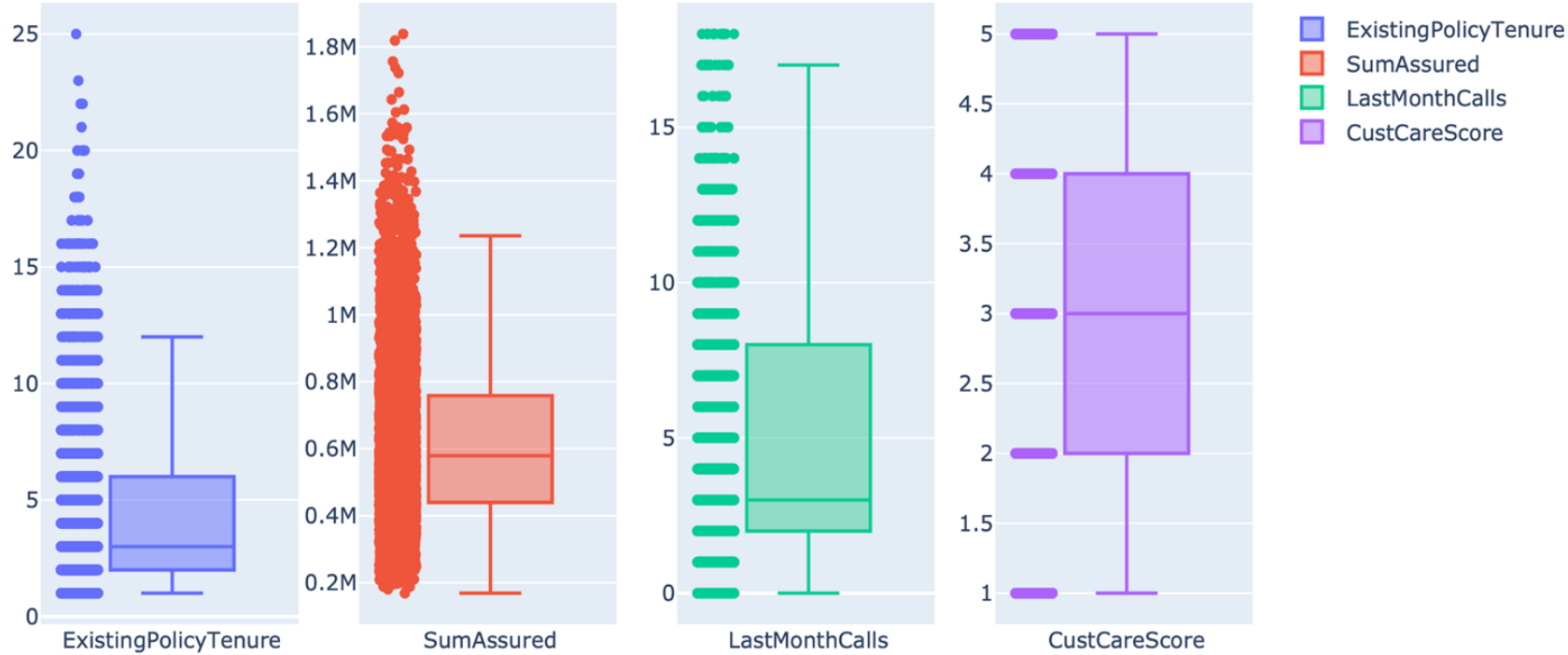
Outliers constitute about 18.1% of our total dataset.

This is a high percentage.

Removing them blindly may tamper the nature of data to larger extent.

So, we need to make sure, if the outliers are authentic data, or faulty numbers, which are making no sense.

Outliers through Visualization



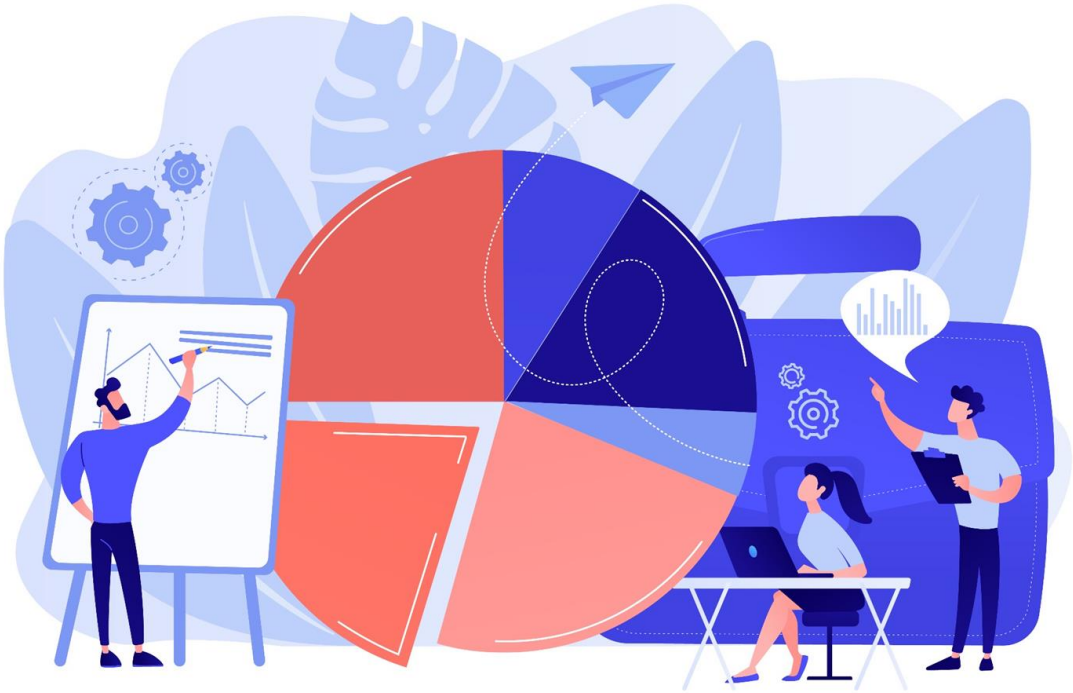
Outliers through Visualization

Outliers

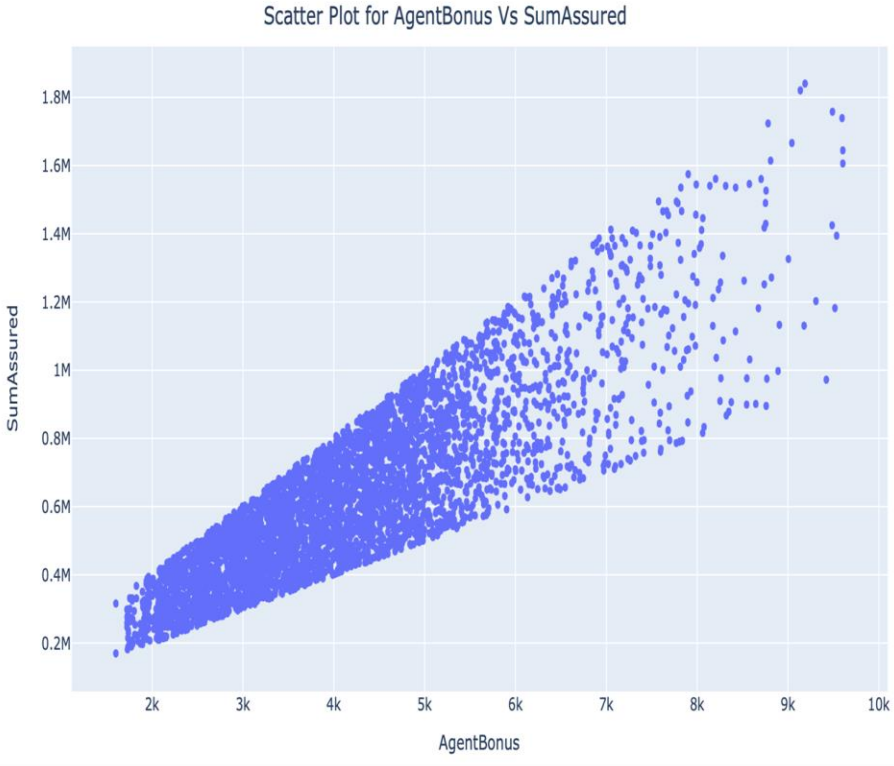
Upon looking at data, we concluded, there are no weird numbers for columns where we have outliers, it's just the nature of data, so we are keeping them.

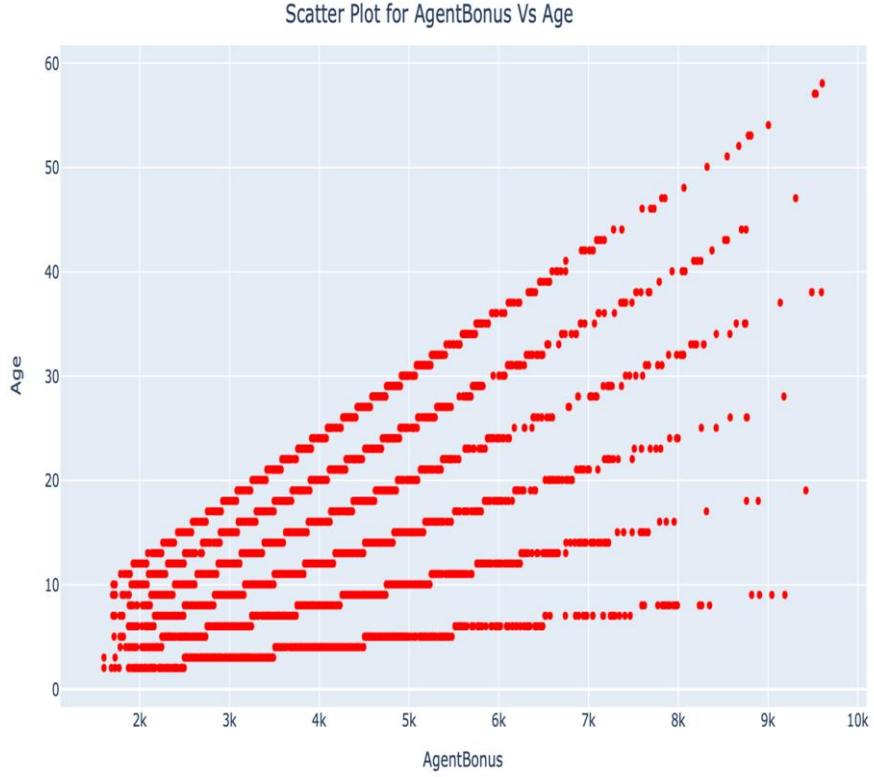
Ex: Few people of age over 40 to 60 are getting life insurance, it's very well within the good range of human life span, so these are authentic data, with no faults. Similarly other columns.

Visualization

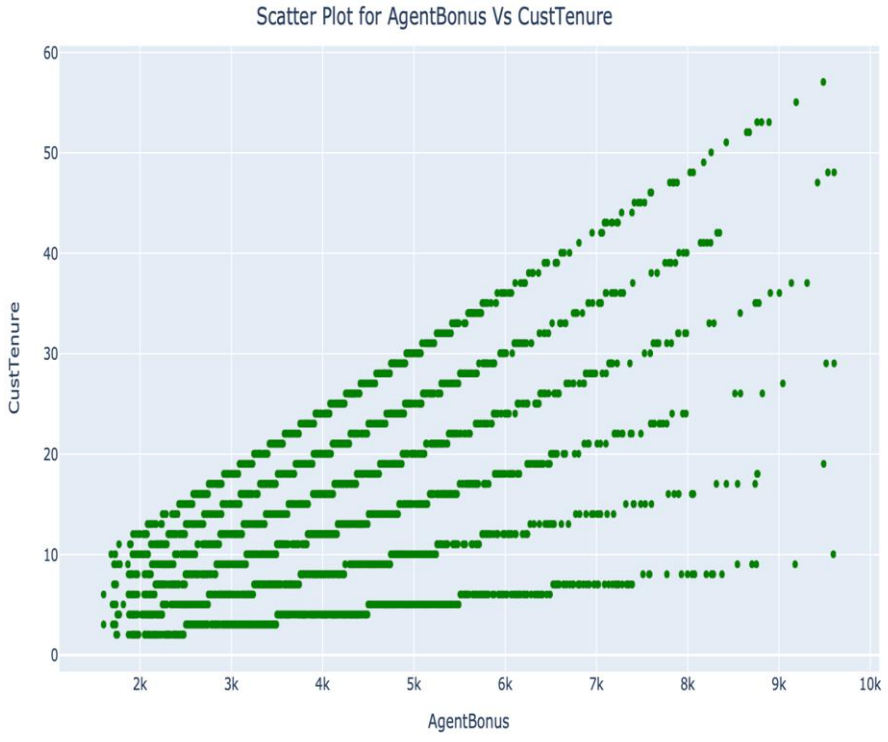


Scatter Plots

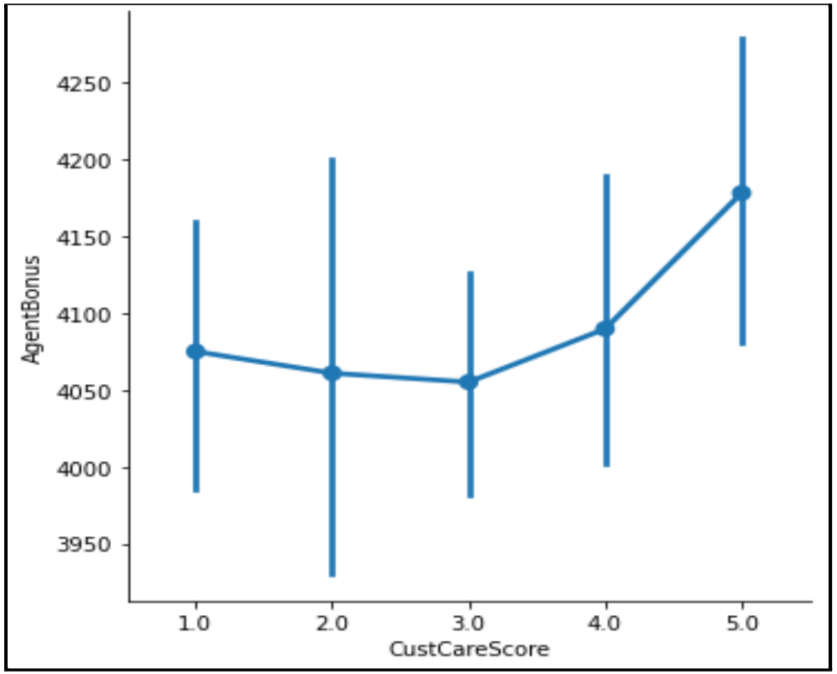




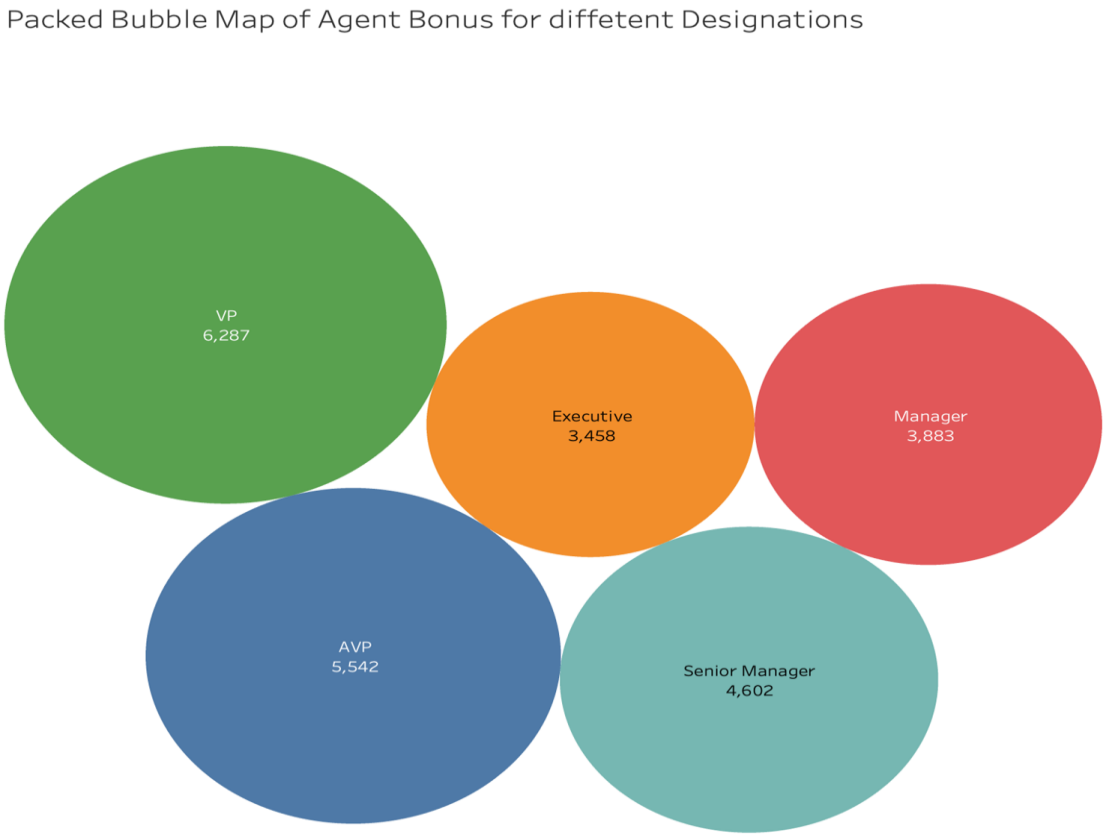
Scatter and Cat Plot



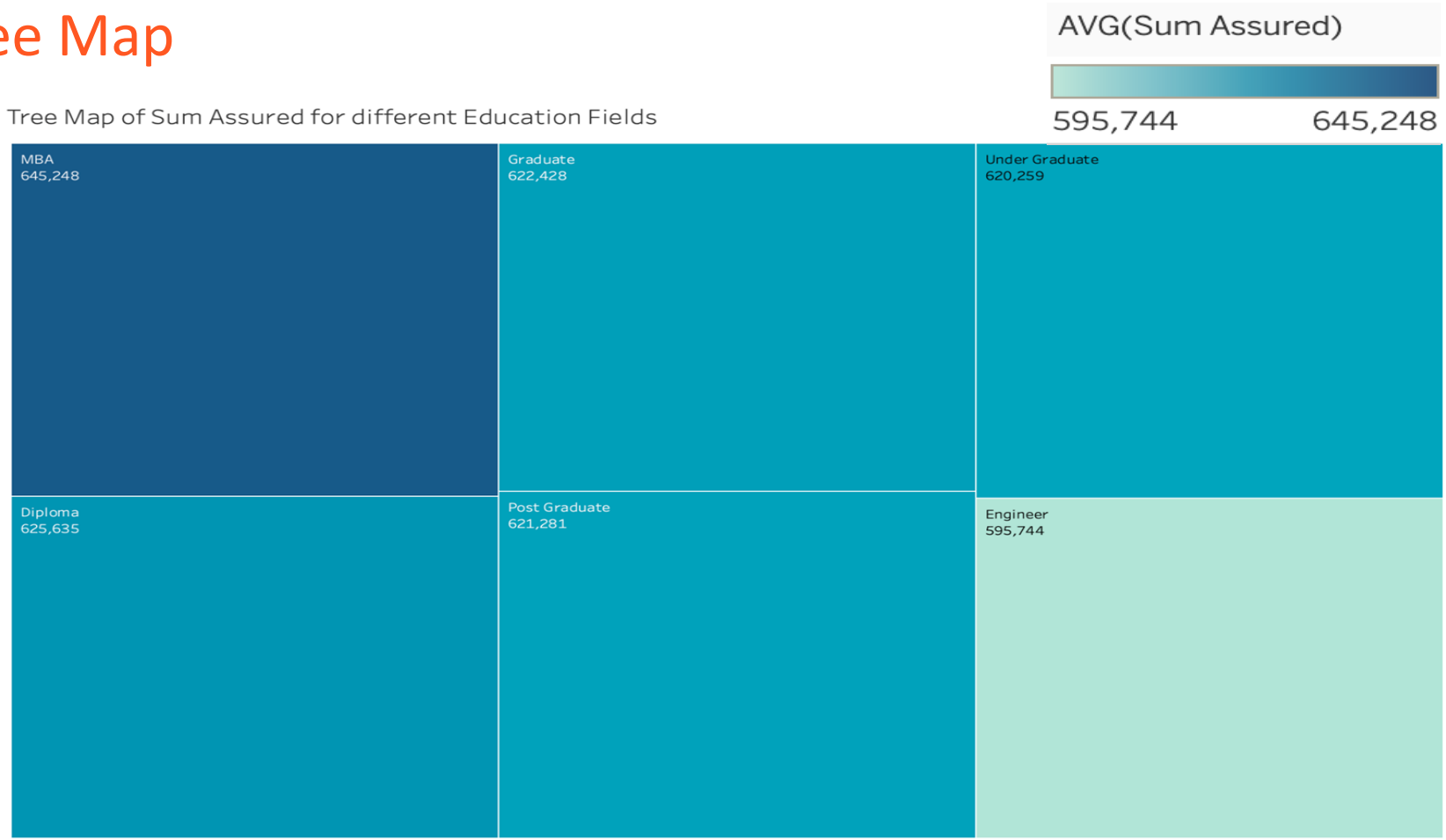
Cat Plot for CustCareScore and Agent Bonus



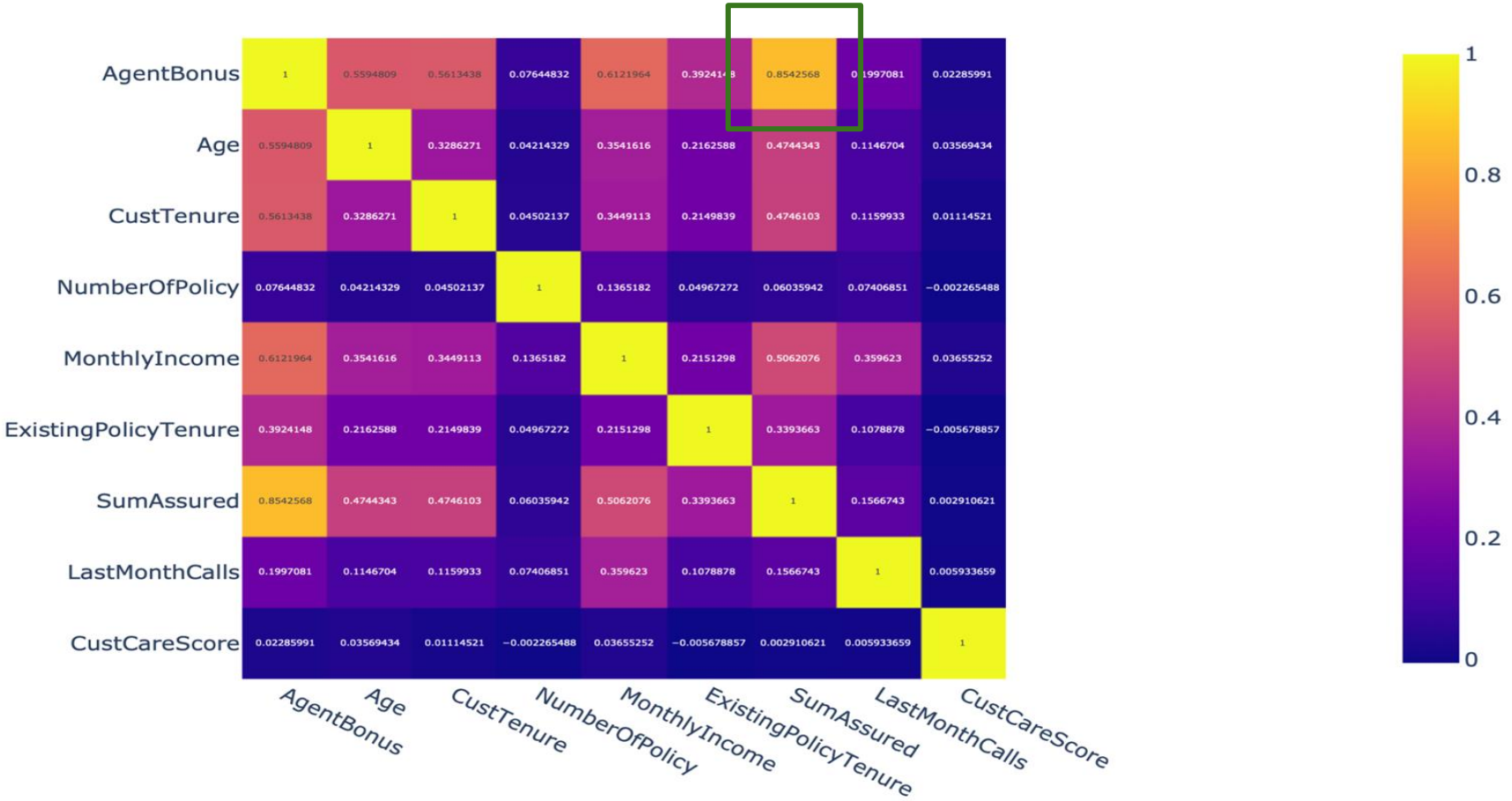
Bubble Map


Packed Bubble Map of Agent Bonus for diffetent Designations
VP
6,287
Executive
3,458
Manager
3,883
AVP
5,542
Senior Manager
4,602

Tree Map


AVG(Sum Assured)
595,744
645,248
Tree Map of Sum Assured for different Education Fields
MBA
645,248
Graduate
622,428
Under Graduate
620,259
Diploma
625,635
Post Graduate
621,281
Engineer
595,744

Correlation Plot



Null values

Column Wise Null Values

Column	# Null Values	% from Total
Age	**69**	**5.9%**
CustTenure	**226**	**5.0%**
NumberOfPolicy	**45**	**1.0%**
MonthlyIncome	**236**	**5.2%**
ExistingPolicyTenure	**184**	**4.1%**
SumAssured	**154**	**3.4%**
CustCareScore	**52**	**1.1%**

Encoding

Used dummy encoding, to encode all categorical columns, also, dropping first column, to minimize dimensionality.

```
pd.get_dummies(cat_cols_df, drop_first=True, dtype=int)
```

After encoding, we are at -

4520 entries and 38 columns.

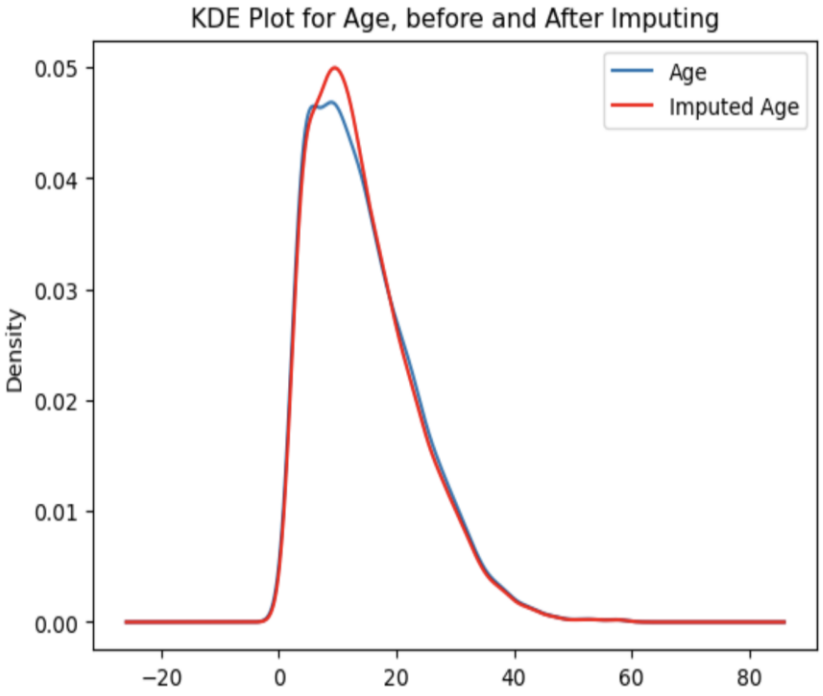
Null Value Imputation

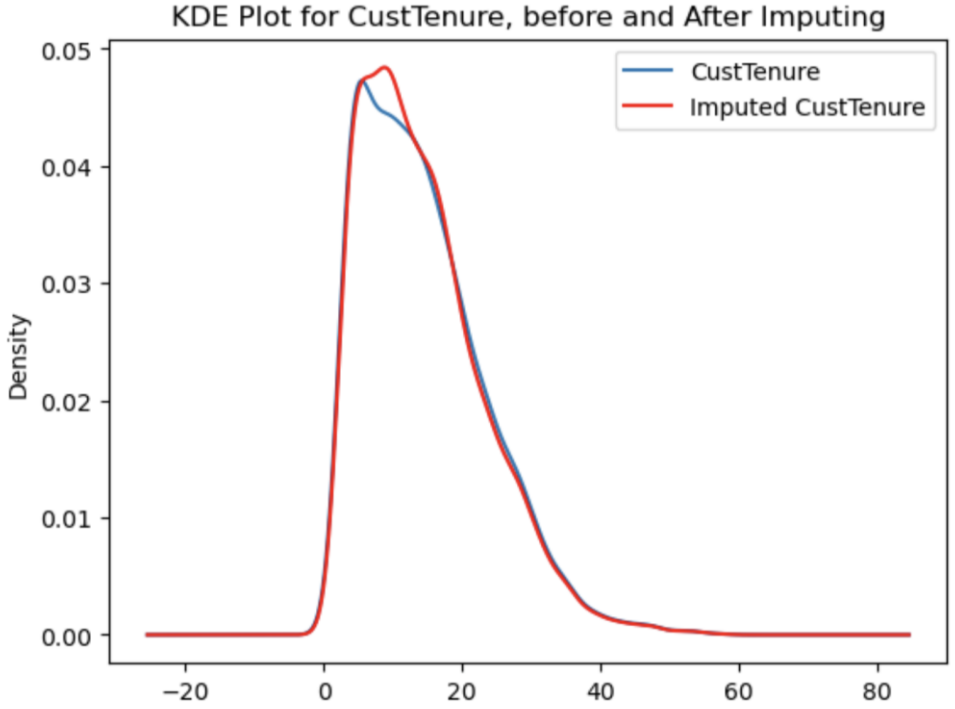
We don’t have any null values in categorical data, but only in numerical data. So, we can encode data, even before imputation and use KNN Imputer.

Method : **KNN Imputer**

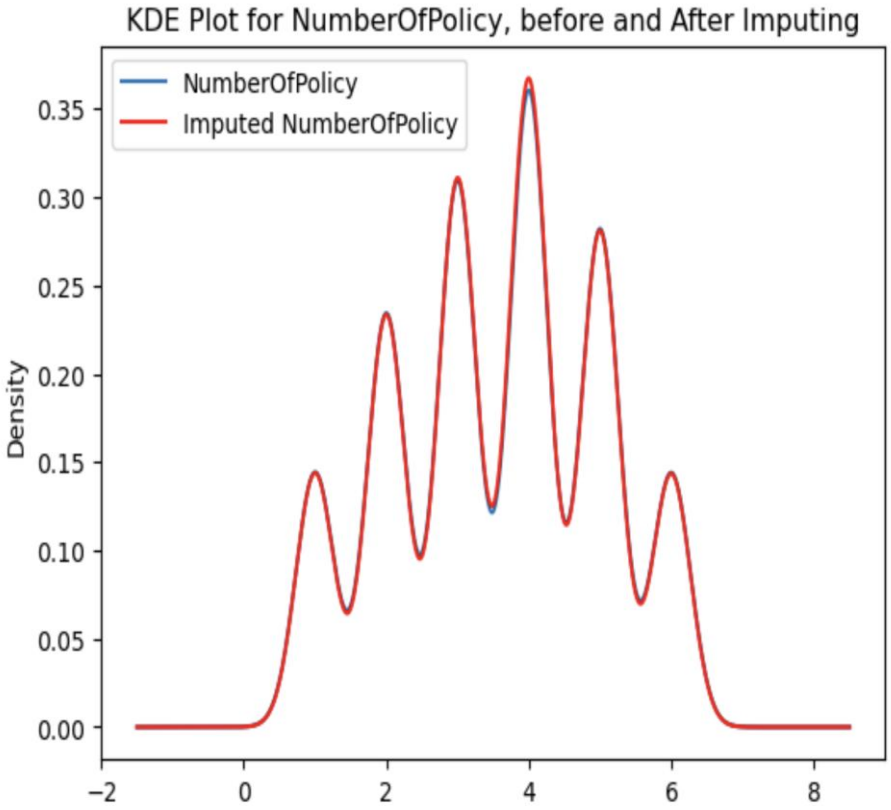
```
from sklearn.impute import KNNImputer

imputer = KNNImputer(n_neighbors=10)

df_imputed = pd.DataFrame(imputer.fit_transform(encoded_df), columns = encoded_df.columns)
df_imputed.head()
```

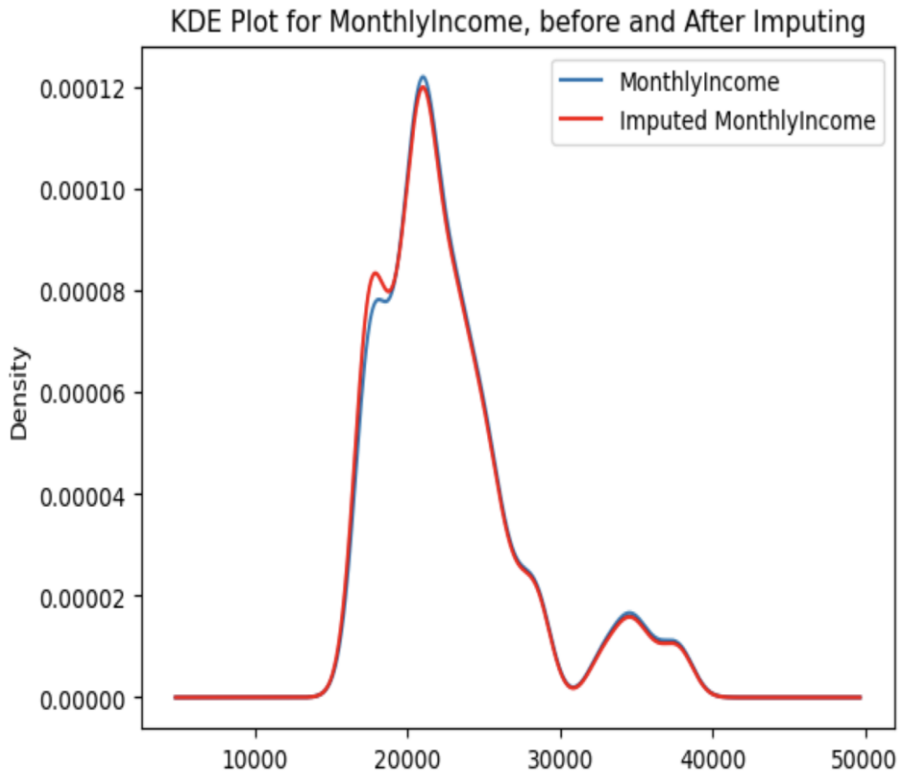
Checking Variance after Imputing 1/4



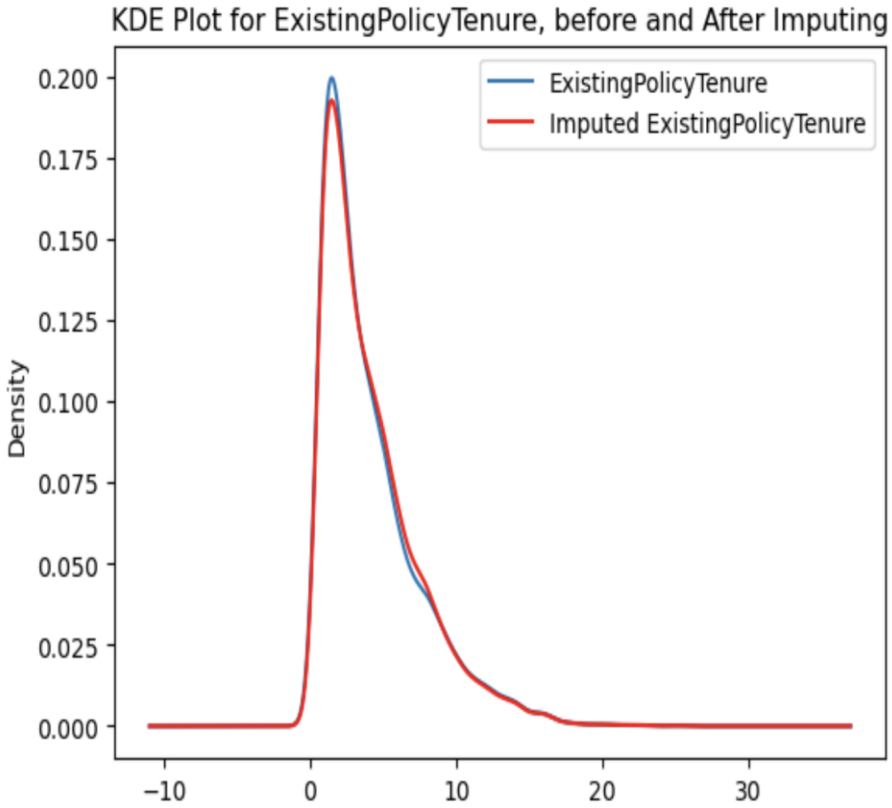


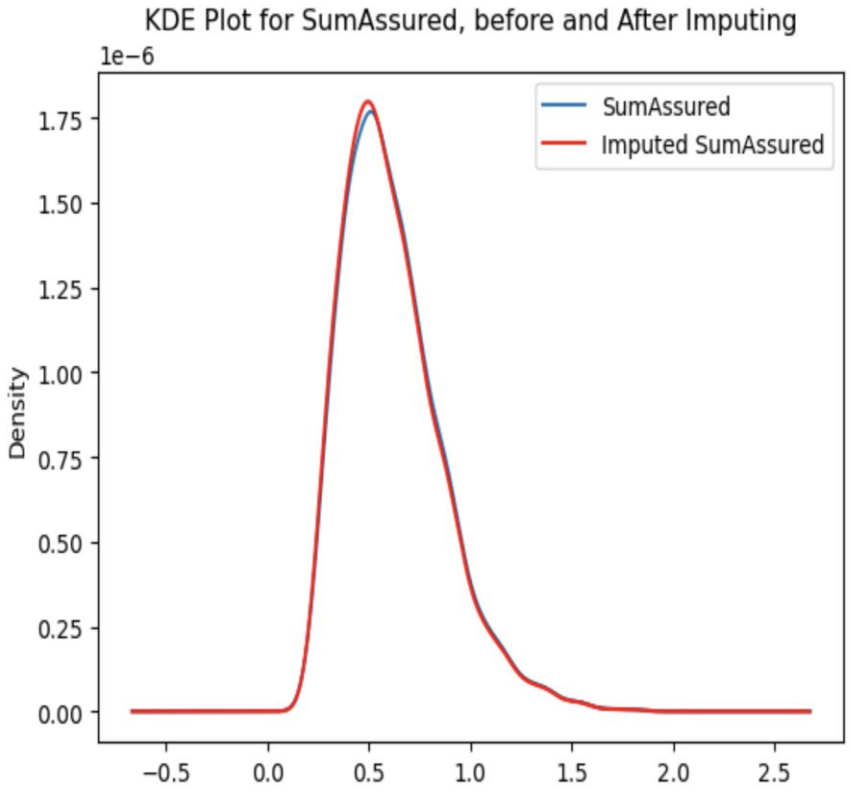
Checking Variance after Imputing 2/4



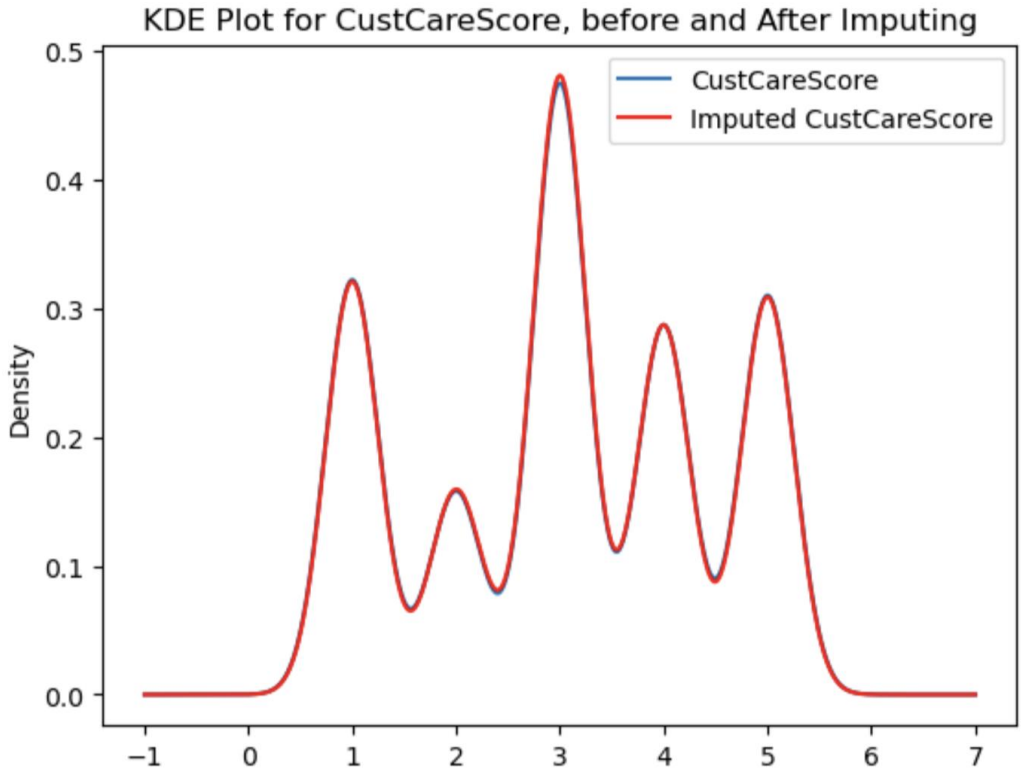


Checking Variance after Imputing 3/4





Checking Variance after Imputing 4/4



Multicollinearity

Removing Multicollinearity with **Variance Inflation Factor (VIF).**

Initially, we had highest VIF as **164**, which is severe multicollinearity

By eliminating one column at a time, with some domain knowledge. We reduced highest VIF number to **< 15.** We had to remove 5 columns out of 38, to get this his score.

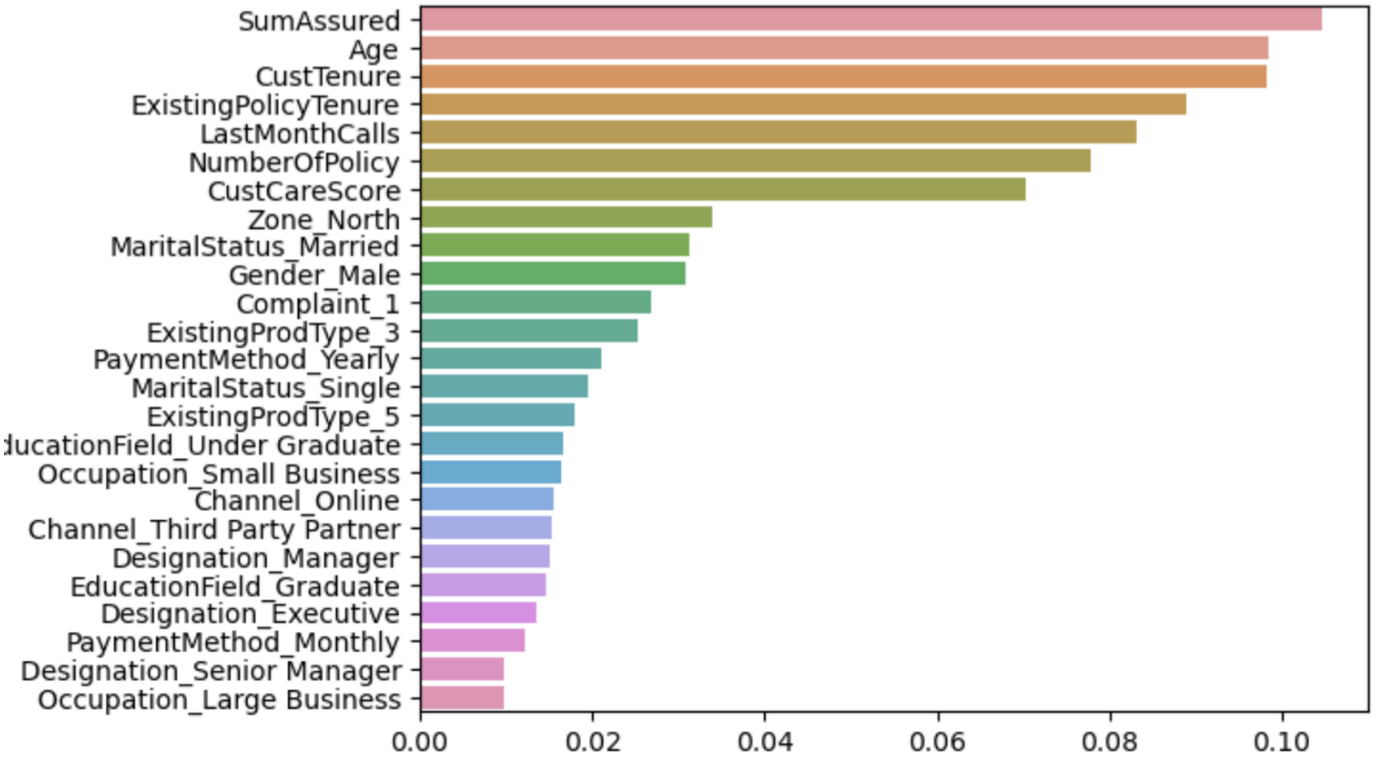
Ex: *Monthly Income* and Education are correlated, with higher levels of education it's natural, they'll earn more. So, one can be removed, so, we don't give the model same information twice.

Feature Selection

```
from sklearn.ensemble import ExtraTreesClassifier
import matplotlib.pyplot as plt

model = ExtraTreesClassifier()
model.fit(X,y)
```

To pick top columns, that are highly significant, by ranking them according to their importance.

Feature Importance



Model Building

1. **Linear Regression - There is fair linear relationship for our target variable with other numerical columns of our dataset. So we can use Linear Regression. We can expect good accuracy already.**

1. **Decision Tree Regressor -** Though we have fair linear relationship with target, still it's not completely linear relation, so, let's check with Decision Tree Regressor, it does a better job at capturing the non-linearity in the data by dividing the space into smaller sub-spaces.

Linear Regression Results

	R2 Train	R2 Test
Base Model	81.7	79
With best features	82.5	81

	RMSE Train	RMSE Test
Base Model	603.2	617.8
With best features	585.0	617.5

Decision Tree Results

Best parameters after hyper parameter tuning
Max_depth = 10
Min_sample_split = 40
Min_samples_leaf = 17

	Score Train	Score Test
Base Model	100	71.6
After Hyperparameter Tuning	87.45	81.73

	RMSE Train	RMSE Test
Base Model	0	731.4
After Hyperparameter Tuning	500.59	586.80

Conclusions

Sum Assured is the biggest factor in determining the Agent Bonus, if agent brings in bigger policy, then bonus will be bigger, also bigger policy tenure and less number of complaints from customers, will increase the chances of getting bigger bonus.

Linear Regression, Decision Tree Regressor models were built, to predict the bonus, an agent can get.

With test accuracies, of **81%** and **82%** respectively.

Thank You!