Engine Rating Prediction

Created by: Keerthi Das

Contents

Data Cleaning Steps

- Converted to appropriate data types
- Imputed nulls according to Meta Data
- Dropped columns that have > 80% nulls
- Dropped columns that have least correlation score (between -0.02 to 0.02)
- All categorical columns were encoded into numbers
- Scaled Data using Standard Scaler

Exploratory Data Analysis

Know your data:

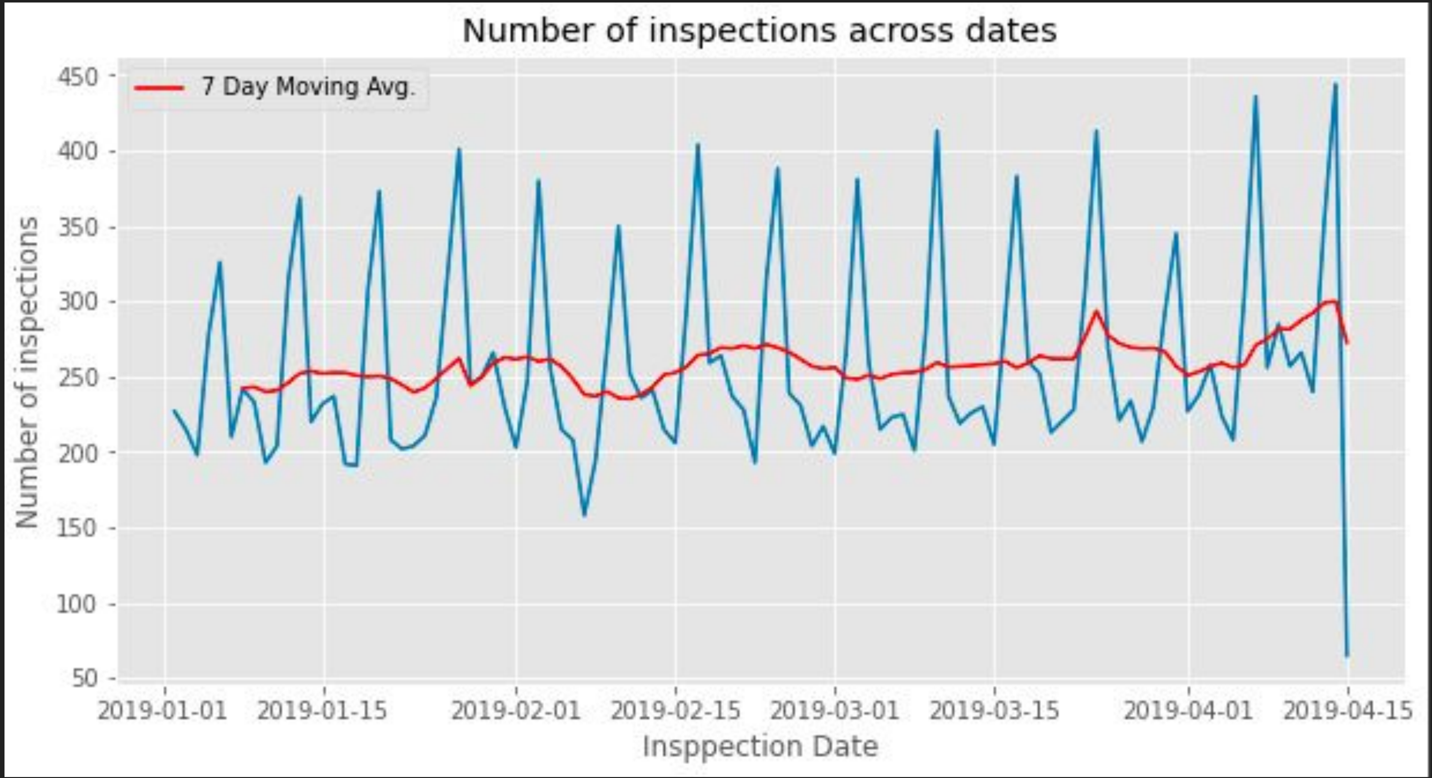
1. There are a total 26307 rows & 73 columns (including target variable).
2. 52 columns/ variables (~72.22%) have more than 40% missing values.
3. Out of these 52 columns 47 are imputed with a constant value of "Yes" which is inferred from the given Meta Data.
4. Rest 5 values were comments which seemed like were given by human experts & hence, are dropped from the set and will not be considered for further analysis.
5. Out of the remaining 68 variables, 63 are of categorical type (~92.6% of total) & 5 are continuous columns (including the target variable).

From the below table, it can be inferred that

1. We have data of vehicles which have registrations from year 1989 to 2019.
2. Minimum reading in the odometer is 1 KM & maximum is 999999.000000 (which seems a little too high, this will be evaluated further).
3. The average vehicle engine rating is 3.62.

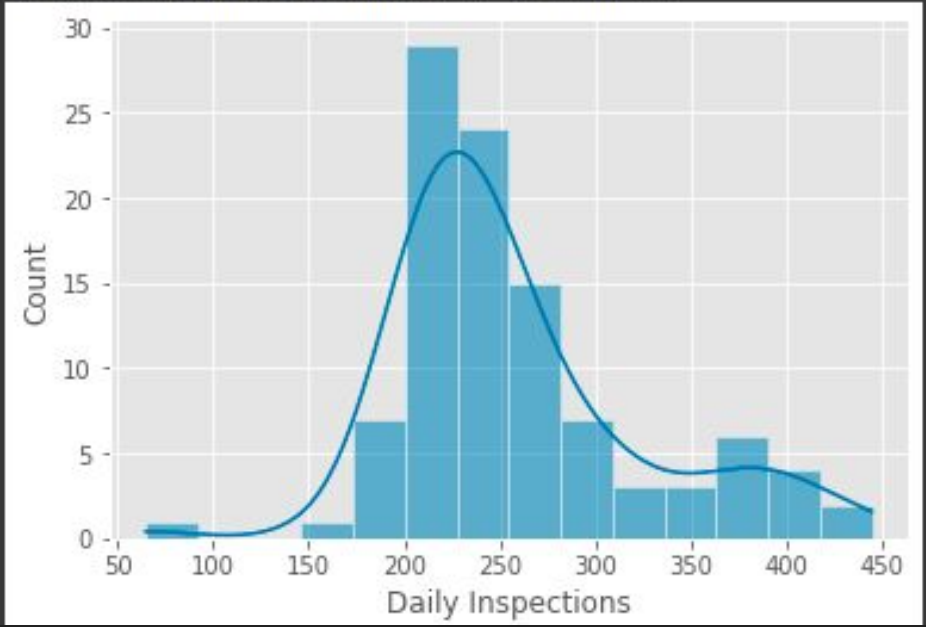
	year	month	odometer_reading	rating_engineTransmission
count	26307.000000	26307.000000	26307.000000	26307.000000
mean	2010.856578	5.462006	76460.143764	3.624663
std	3.766234	3.583866	46762.524489	0.847645
min	1989.000000	1.000000	1.000000	0.500000
25%	2008.000000	2.000000	46396.000000	3.500000
50%	2011.000000	5.000000	72013.000000	4.000000
75%	2014.000000	9.000000	98289.500000	4.000000
max	2019.000000	12.000000	999999.000000	5.000000

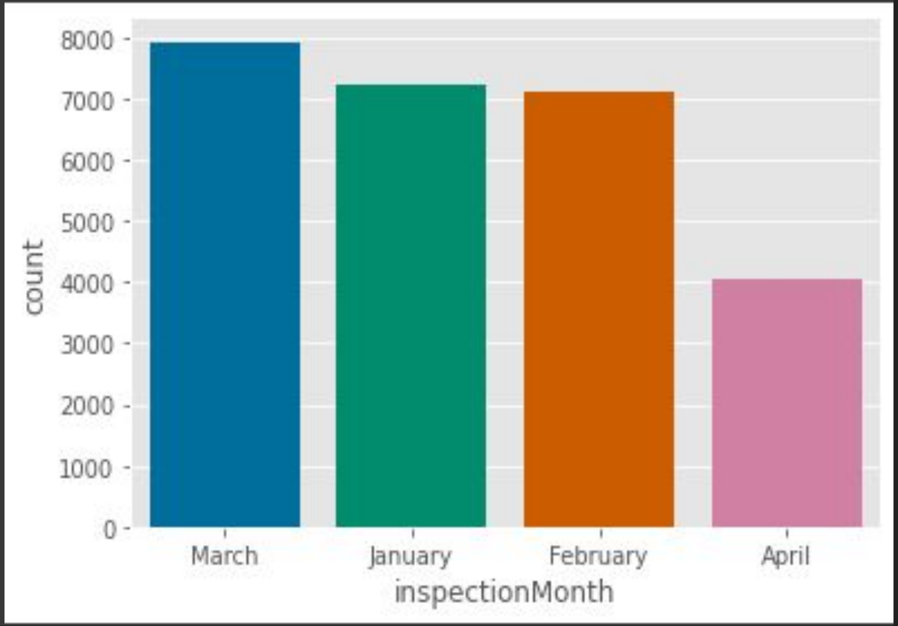
Univariate Analysis



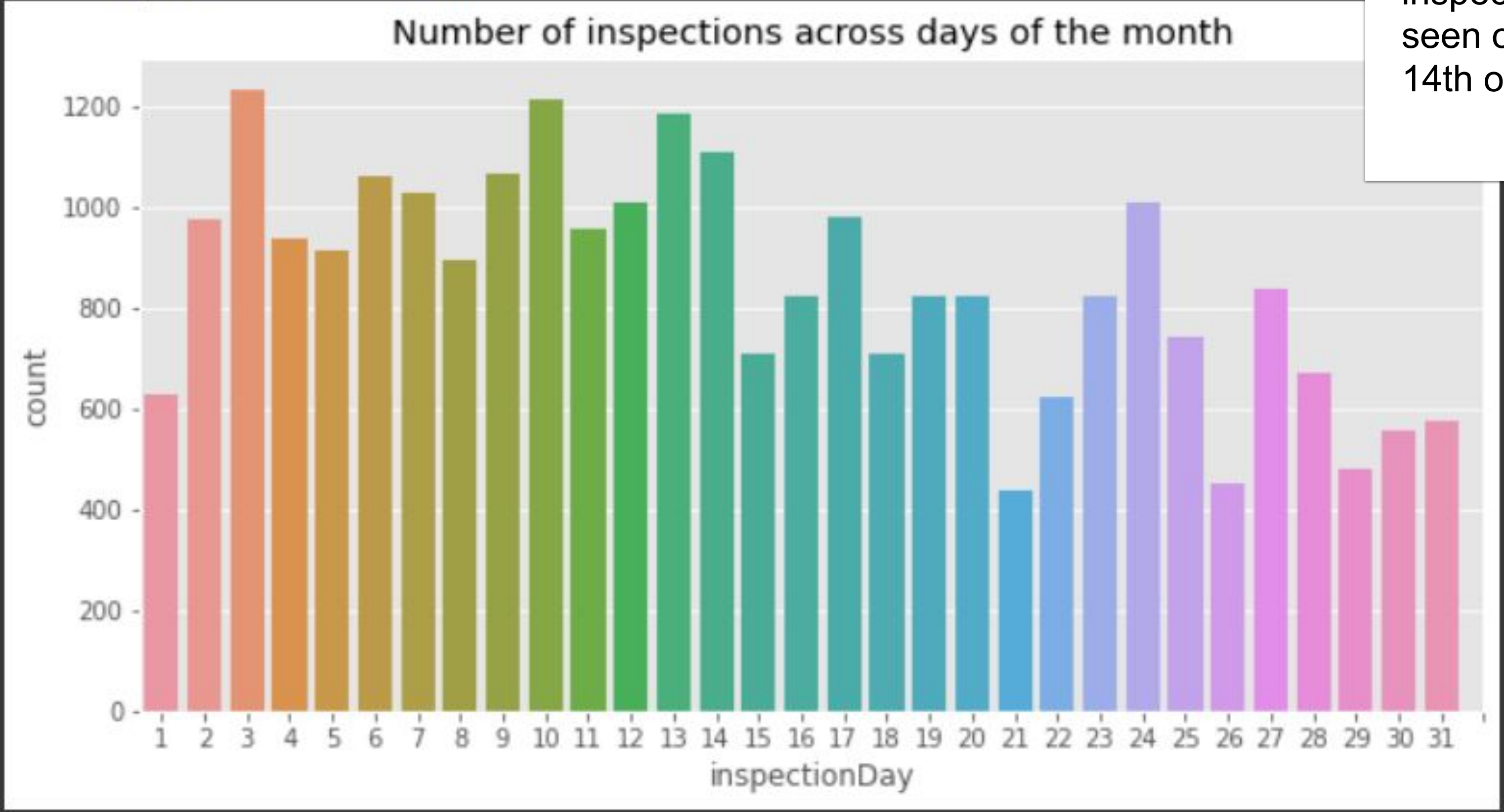
March seems to have highest number of inspections

```
Average daily inspections (for this sample): 257.9117647058824
We have inspection data for 102 dates
```





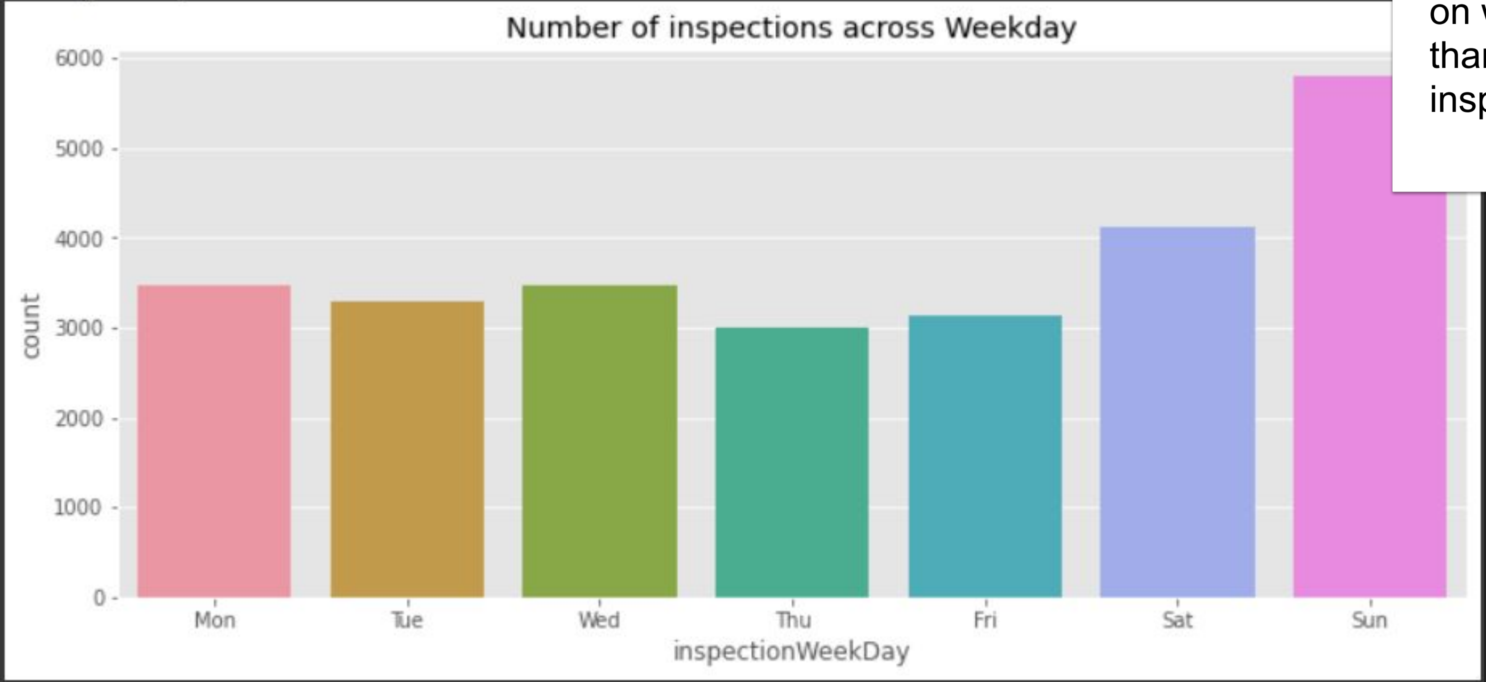
	Months	Distribution(%)
0	March	30.125062
1	January	27.426160
2	February	27.011822
3	April	15.436956



Key findings:

Few high spikes of inspections can be seen on 3rd, 10th, & 14th of the month.

```
inspectionWeekDay
0    3463
1    3302
2    3468
3    3011
4    3137
5    4120
6    5806
Name: appointmentId, dtype: int64
Average Weekly Inspections: 3758.1428571428573
Average inspections on Weekends: 4963.0
```

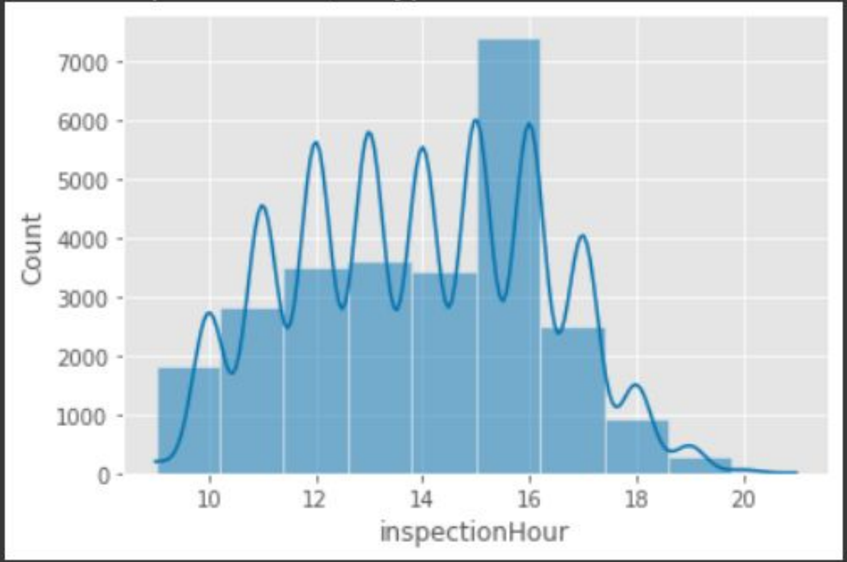


Key findings:

Clearly, there are more inspections on Weekends.

Average inspections on weekend is greater than weekly average inspections by 32%.

```
15    14.167332
16    14.000076
13    13.661763
12    13.251226
14    13.026951
11    10.719580
17     9.522180
10     6.431748
18     3.527578
19     1.098567
9      0.440947
20     0.136846
21     0.015205
Name: inspectionHour, dtype: float64
```



DATE TIME SUMMARY:

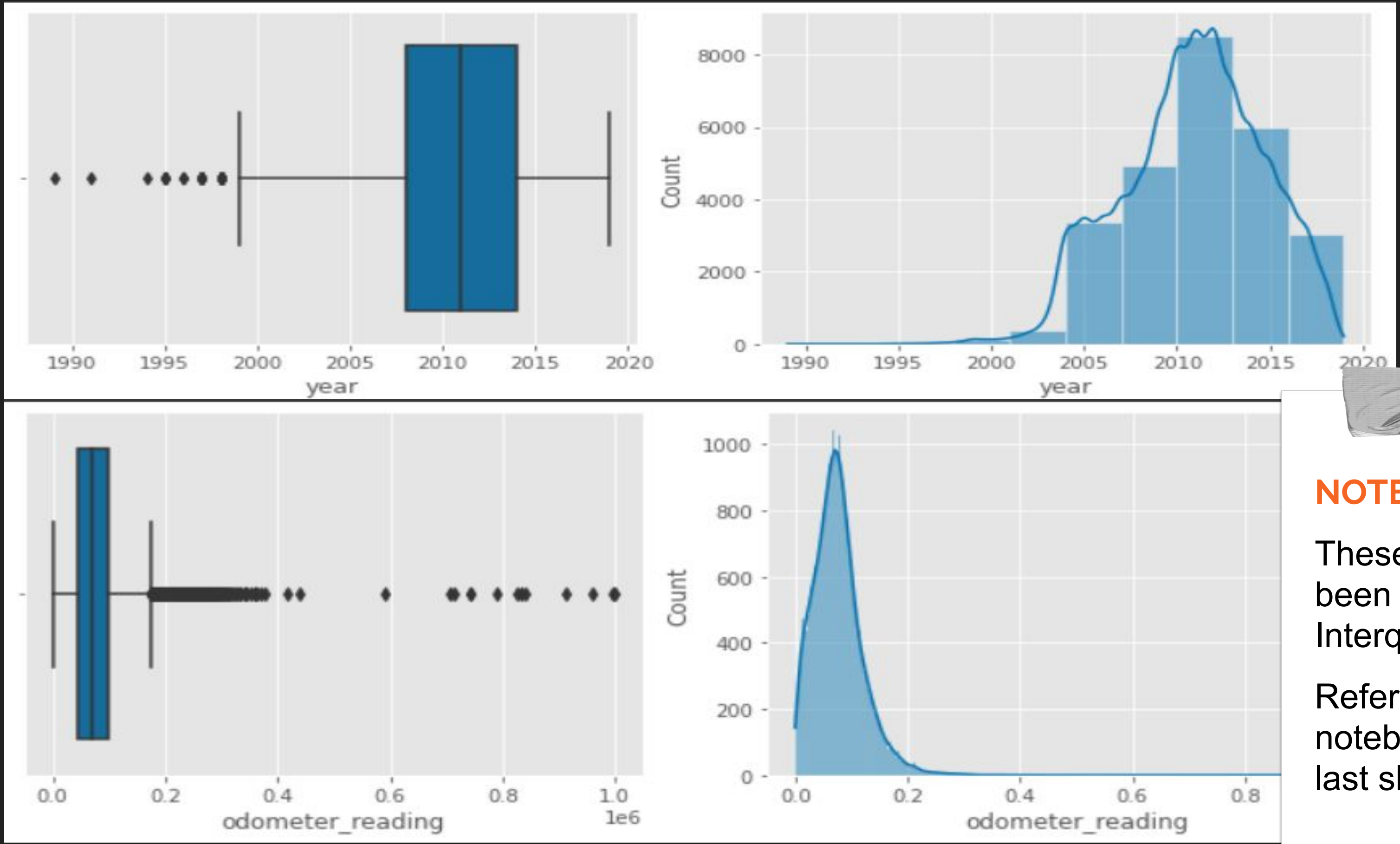
1. Monthly representation is maximum for March (~ 30%) followed by January (~ 27.4%), February(~ 27%) & April (~ 15.4%).

2. High spikes in inspections can be seen on 3rd, 10th, & 14th of the month.

3. Average inspections on weekend is greater than the overall weekly average inspections by ~ 32 %.

4. There's a spike in number of inspections across from 15:00 - 16:00.

Year & Odometer Reading seems to have few outliers



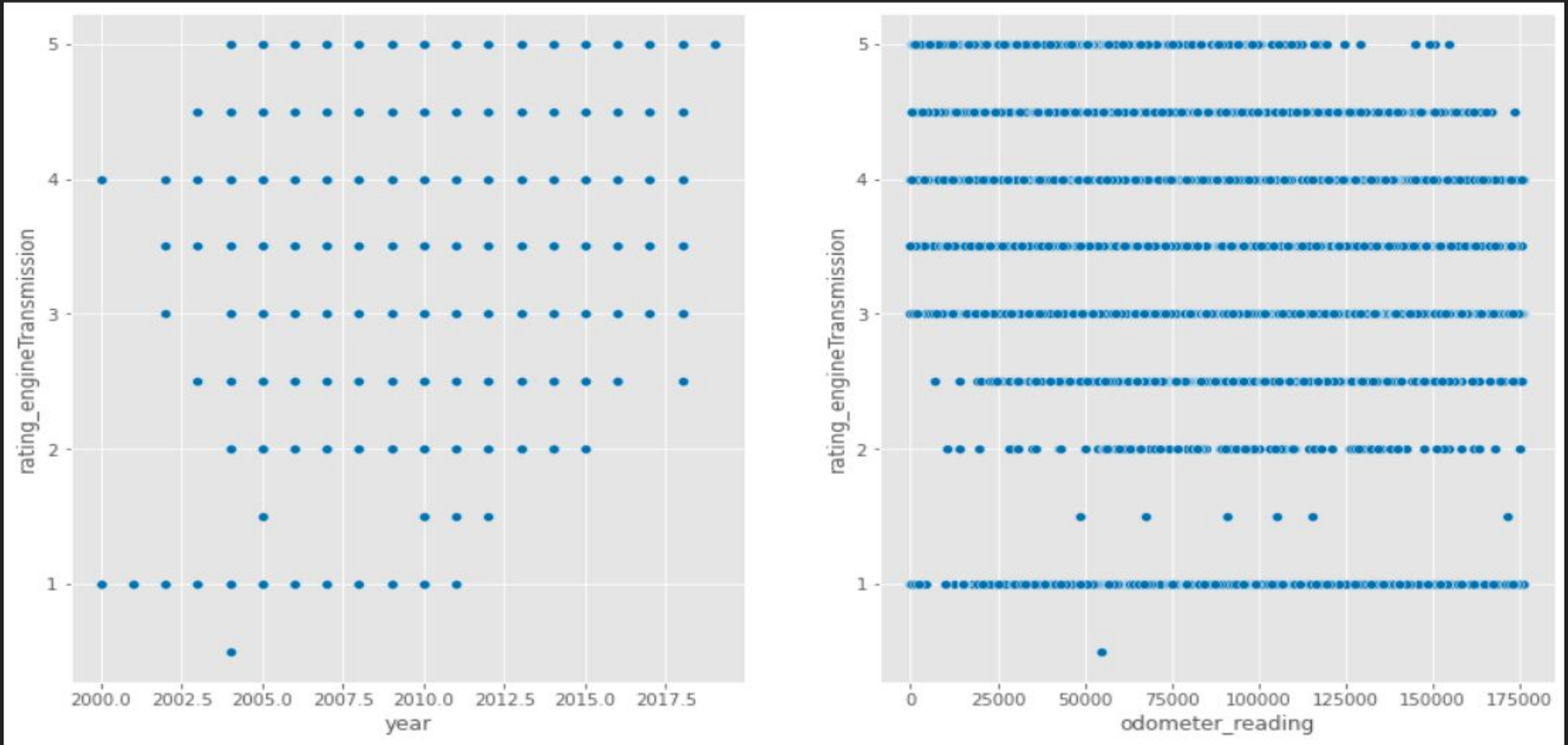
NOTE:

These outliers have been treated using Interquartile Range.

Refer to the colab notebook (shared in last slide) to know how.

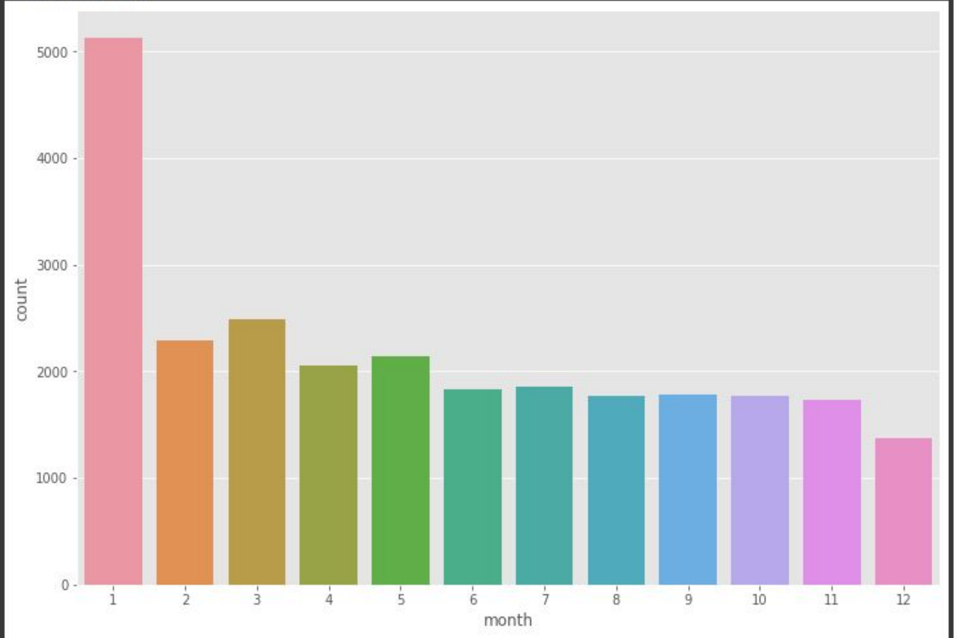
Bivariate Analysis

Target Variable (Engine Rating) seems to be more of a discrete variable.

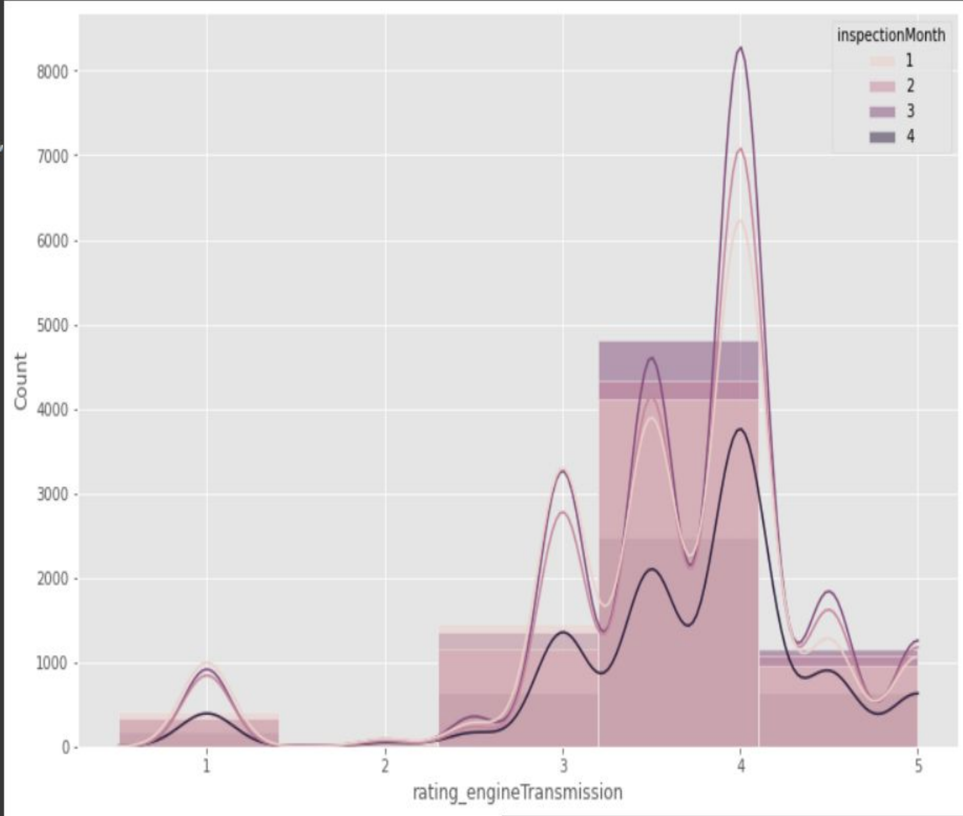


Distribution of Registration month

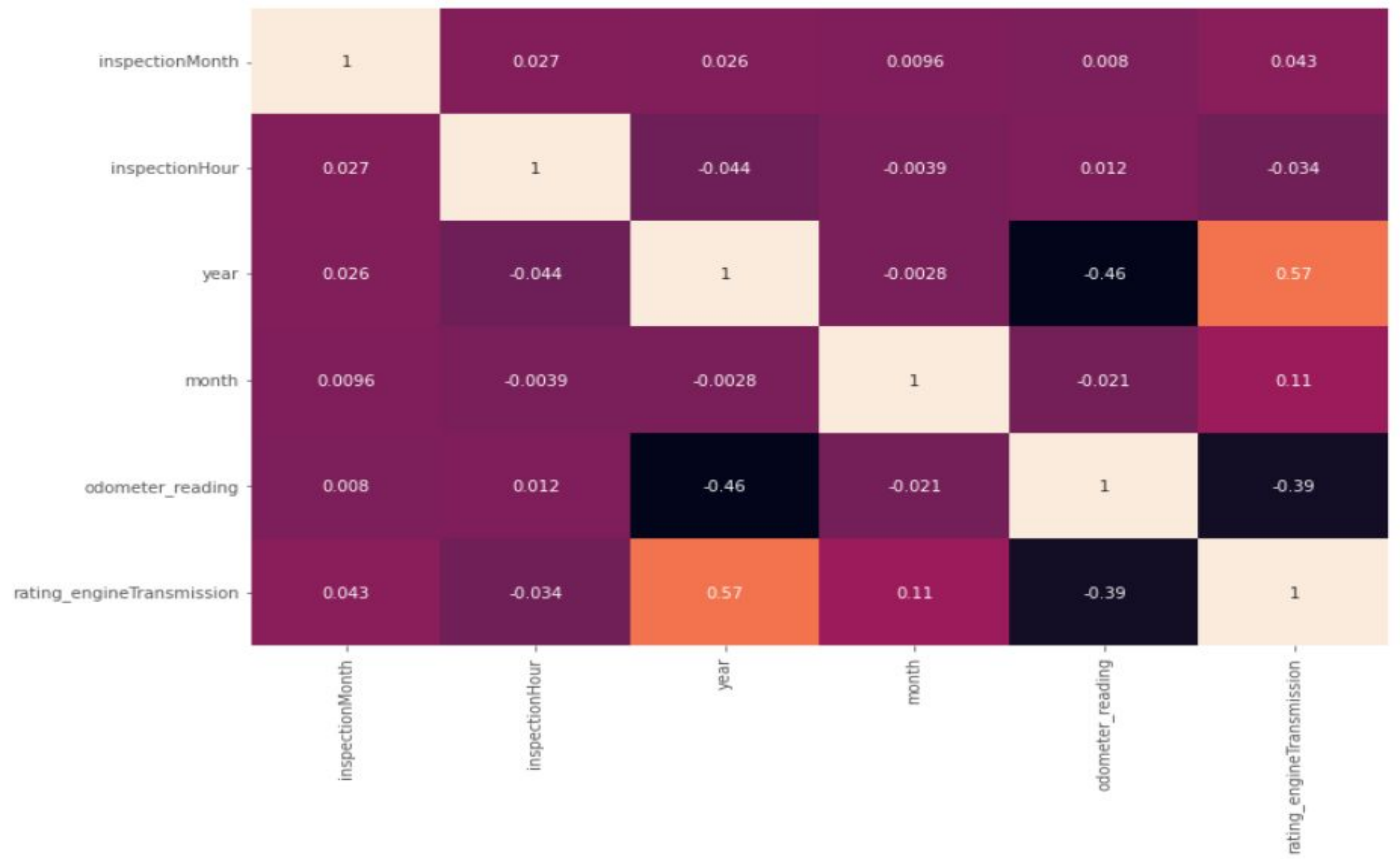
```
   Registration_Month  Distribution{%}
0                 Jan        19.568367
1                 Mar         9.475330
2                 Feb         8.735606
3                 May         8.186532
4                 Apr         7.850988
5                 Jul         7.092199
6                 Jun         6.966369
7                 Sep         6.783345
8                 Aug         6.768093
9                 Oct         6.756654
10                Nov         6.588881
11                Dec         5.227637
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the follow
  FutureWarning
```



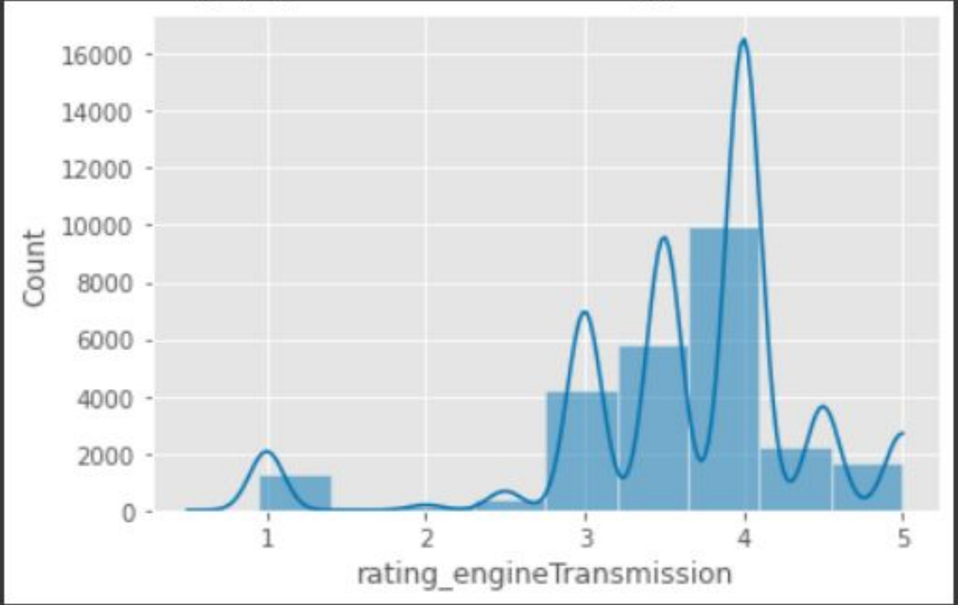
Inspections by Month



Year (registration year) & Odometer_reading are negatively correlated with Engine Rating



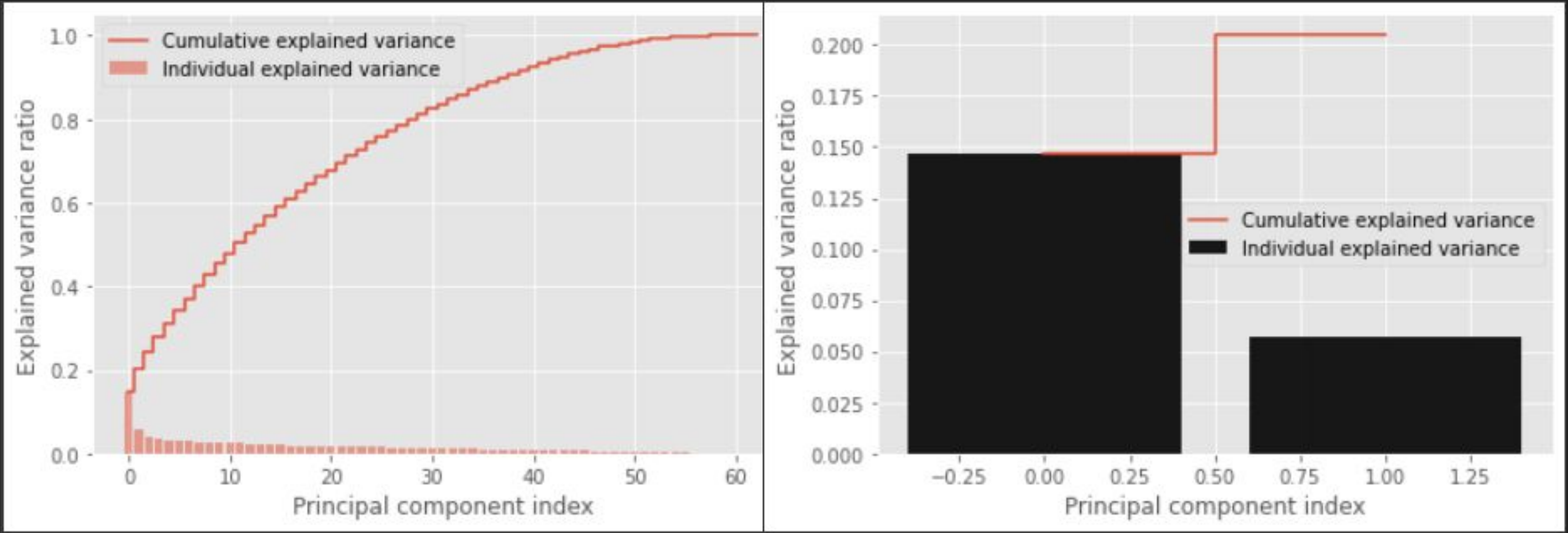
```
4.0    39.033472
3.5    22.634632
3.0    16.469389
4.5     8.603120
5.0     6.353375
1.0     4.863996
2.5     1.575606
2.0     0.438975
1.5     0.023517
0.5     0.003919
Name: rating_engineTransmission, dtype: float64
```



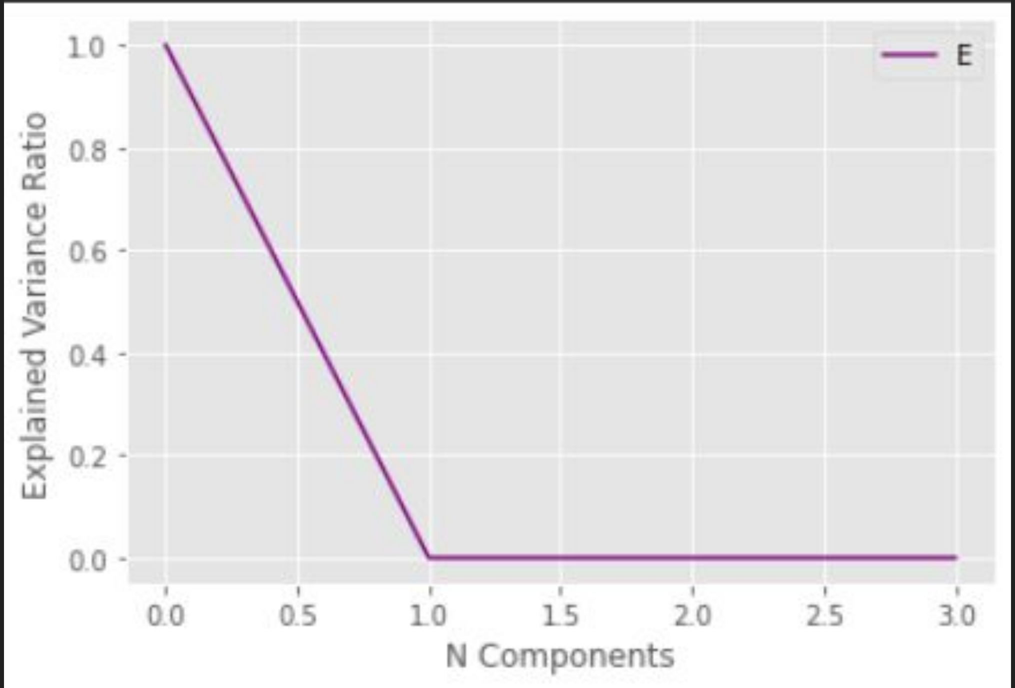
Analysing Target i.e. Engine Rating:

1. Engine Rating of 0.5, 1.5 & 2.0 have a very low count (less than 1%).
2. The rating_engineTransmission appears to be more of a discrete variable.
3. Rating is between 0 to 5.
4. Average vehicle engine rating is 3.62.
5. Rating is negatively correlated with Year & Odometer Reading.

Dimensionality Reduction: Principal Component Analysis



Validation Using Elbow Method



Insights on PCA:

1. According to previous slide, nearly 50% variability is explained by 5 components alone.

2. Of this 50%, 25% variability is explained by 2 principal components alone.

3. This is validated using Elbow Method. The line sharply falls down between 1 to 2.

4. Hence, I will be taking 2 principal components.

Models that were trained & Accuracy comparison:

Model	Train Accuracy	Test Accuracy	
Multiple Linear Regression	0.405	0.406	Poor Accuracy
Polynomial Regression	0.176	0.177	Poor Accuracy
Decision Tree Regression	1.0	1.0	Impractical
Random Forest Regression	0.928	0.504	Overfitting

Techniques used to improve Accuracy

Technique	Train Accuracy	Test Accuracy
Bagging	0.959	0.709
Extreme Gradient Boost	0.690	0.680
Light Gradient Boost	0.772	0.726

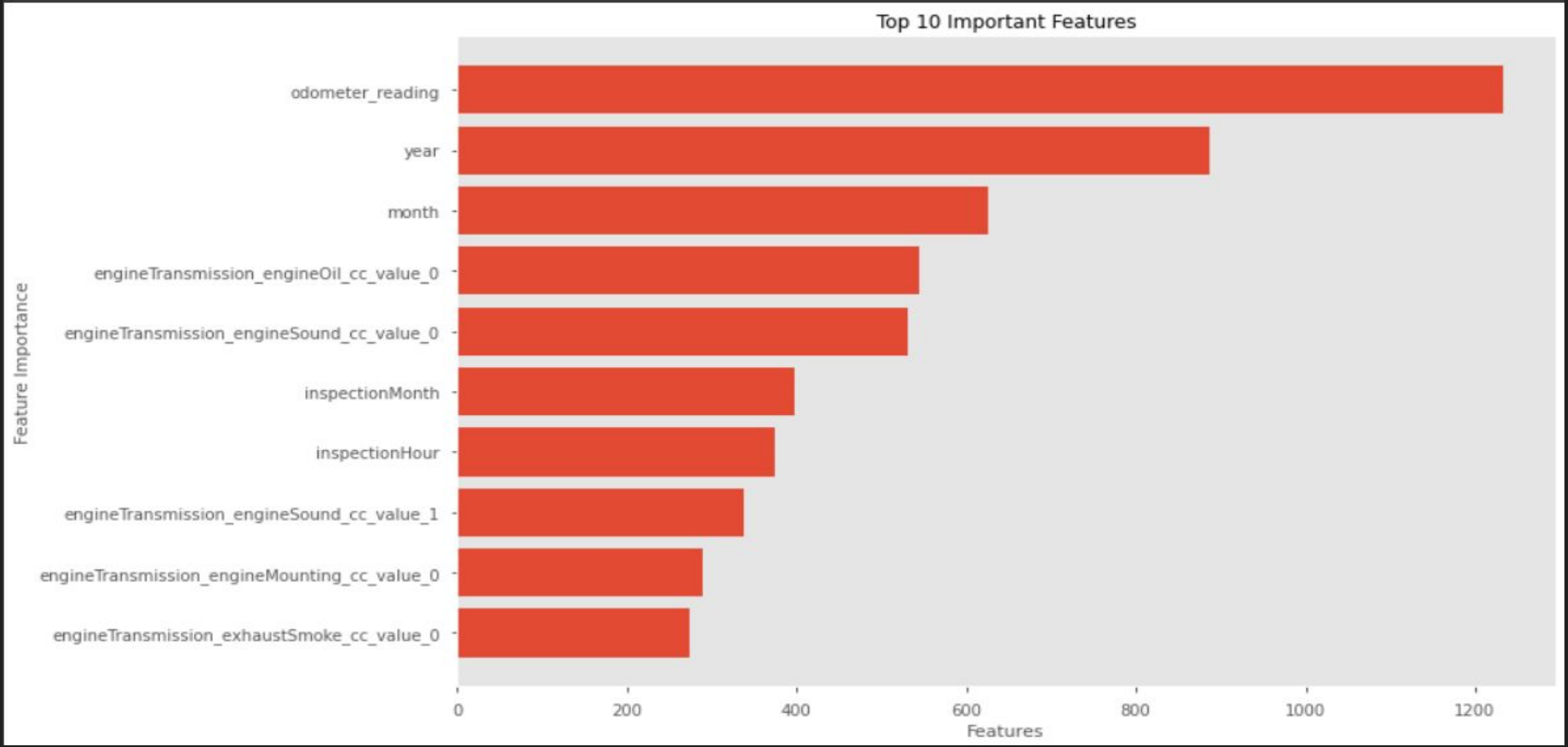
Overfitting

Still not satisfactory

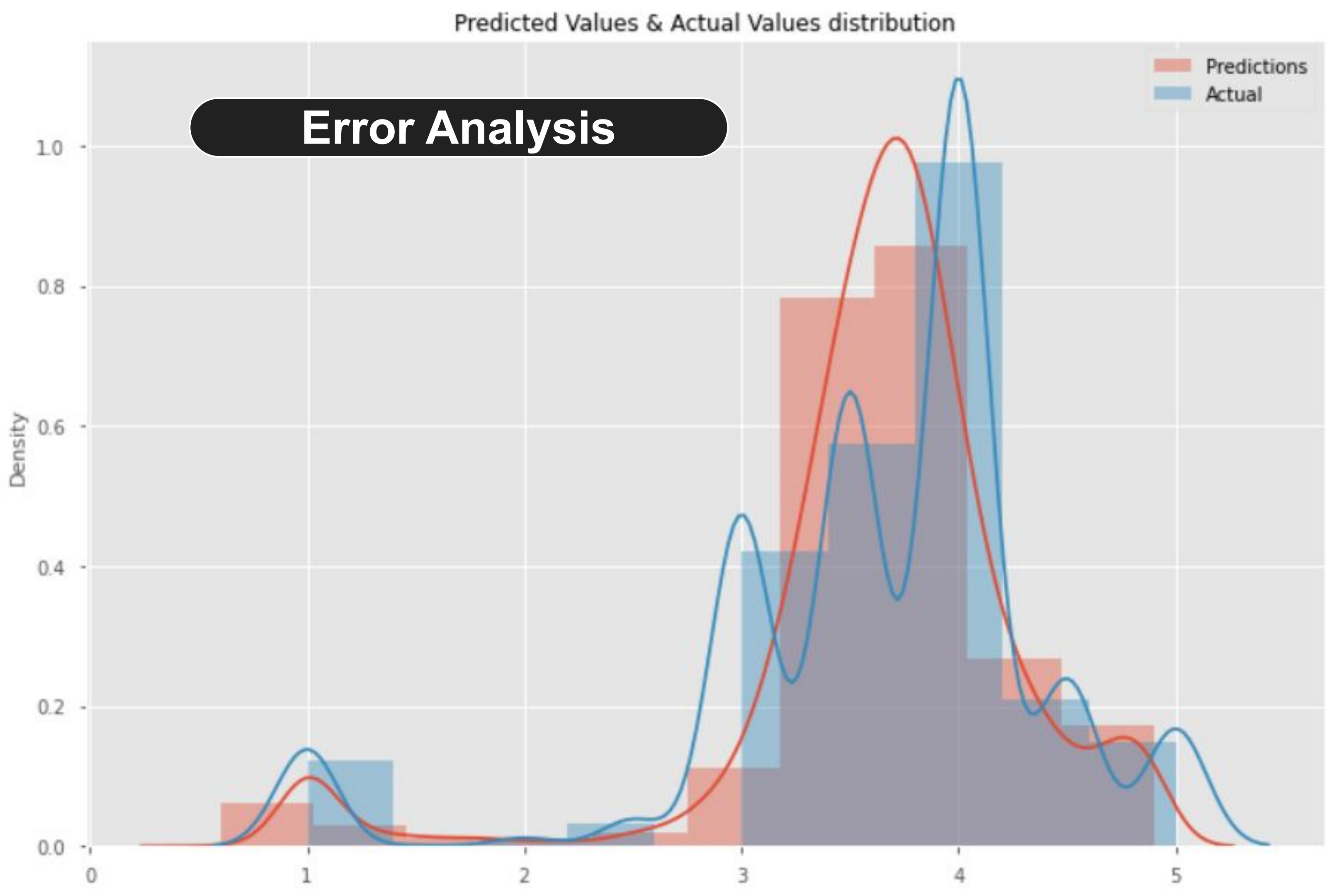
Accuray improved. We’ll use Grid Search CV to find best parameters & further optimise it

Post using Grid Search CV

Final Train Accuracy	Final Test Accuracy
0.788	0.729



Error Analysis



Good luck!

I hope you'll use these insights to go out and deliver a memorable pitch for your product or service!

To access the python notebook used for this analysis, please click here:

https://colab.research.google.com/drive/1YhiB-p38yGDiDGOMSfmX39aGS-QmKyIF?usp=sharing

Dataset:
https://docs.google.com/spreadsheets/d/19K9M9Sg78tCpyJnb-aOOI9yJQxlv7R6wlkW6O2ziA1E/edit#gid=0