Project Report

Contents

[Overview 3](#_Toc98259725)

[Potential Users 3](#_Toc98259726)

[Project Description 4](#_Toc98259727)

[Problem Case 4](#_Toc98259728)

[Initial Exploration 4](#_Toc98259729)

[Data Search 5](#_Toc98259730)

[Data Collection 6](#_Toc98259731)

[Data Description 8](#_Toc98259732)

[Data Cleaning 9](#_Toc98259733)

[Stipend 9](#_Toc98259734)

[Duration 10](#_Toc98259735)

[Location 11](#_Toc98259736)

[Title 13](#_Toc98259737)

[Applicants 15](#_Toc98259738)

[Skills Required 18](#_Toc98259739)

[Perks 19](#_Toc98259740)

[Number of Openings 19](#_Toc98259741)

[Data Exploration 19](#_Toc98259742)

[Skills Required 19](#_Toc98259743)

[Perks 20](#_Toc98259744)

[Company 21](#_Toc98259745)

[Title 22](#_Toc98259746)

[Location 23](#_Toc98259747)

[Duration – Stipend Relationship 24](#_Toc98259748)

[Number of Openings – Stipend Relationship 26](#_Toc98259749)

[Target Variable Transformation 28](#_Toc98259750)

[Log Transformation 29](#_Toc98259751)

[Box-Cox Transformation 30](#_Toc98259752)

[Square Root Transformation 31](#_Toc98259753)

[Creating Dummy Variables 32](#_Toc98259754)

[Pre-Modelling Tests 33](#_Toc98259755)

[One-Way ANOVA 33](#_Toc98259756)

[Pearson Correlation 33](#_Toc98259757)

[Variance Inflation Factor (VIF) 34](#_Toc98259758)

[Model Building 34](#_Toc98259759)

[Selection of Model 35](#_Toc98259760)

[Selection of Transformed Variable 35](#_Toc98259761)

[Hyper-Parameter tuning of the Model 35](#_Toc98259762)

[Regularization 35](#_Toc98259763)

[Randomized Search CV 37](#_Toc98259764)

[Feature Importance 37](#_Toc98259765)

[Saving of the Model 38](#_Toc98259766)

[Productionizing the Model 38](#_Toc98259767)

# **Overview**

* Created an app that estimates Internship stipends on the basis of personalized factors for each individual user with inputs ranging widely from location, skills, perks, duration, etc.
* Scraped over 15000+ internships from Internshala.com using web scraper written on Python.
* Cleaned and manipulated the entire data extensively on Python to make it usable.
* Engineered new features and performed deep exploration on the data.
* Performed various pre-modelling statistical tests to understand the data better.
* Built a machine learning model with the best hyper-parameters.
* Built a client facing API using flask from scratch and hosted the app online for anyone to use.

## Potential Users

* Students interested in knowing the benefits and the amount of stipend they should expect when applying for internships.
* Companies / Organizations / Firms while deciding how much they should offer as stipend when issuing new internship positions.

# **Project Description**

## Problem Case

The main idea behind the project came from when I was searching for internships online. I realized that there was no practical way to get an estimate for the stipend one should expect based on the skills the person has when looking for internships.

I also realized that there are many other factors like location, internship duration, perks the user wants, flexibility of work, etc. that could play a role in deciding the stipend amount.

**But what got me really curious was the question that do these factors really play a role in deciding a stipend amount?**

**If yes, can we build a machine learning model around it that can estimate the stipend amount for an end user?**

## Initial Exploration

When searching through online websites, I found many estimates from all around the world but none provided a personalized estimation based on related factors.

Glassdoor.com Internship Estimation

Graphical user interface, application, email

Description automatically generated

The information I found was generally vague and varied widely when compared across all the internships that were on offer in the market.

## Data Search

I proceeded to search for datasets on Indian internship markets that could provide me the raw information I needed to explore and extract information from. But I couldn’t find any useful data whatsoever.

Then, I decided to scrap the data myself from Internshala.com, which is one of the largest internship platforms in India.

# Data Collection

For my data collection, I needed to write a web scraper application for Internshala.com website.

With a quick google search, I found that someone had already written web scraper application for Internshala.com two years ago.

[GitHub Link of Web Scraper](https://github.com/het-parekh/Internshala-Web-Scraper-Internshala.com)

(The owner can be contacted on [hetparekh26@gmail.com](mailto:hetparekh26@gmail.com) for any inquiries.)

The code was outdated and was not working as expected due to changes in the website over the years. But it provided a good starting point.

With some modifications, tweaking and debugging, I was able to make the code work successfully. After running the scraper overnight, we got the dataset with the following features:

* Title
* Company
* Location
* Duration
* Stipend
* Apply By
* Applicants
* Skills Required
* Perks
* Number of Openings
* Link

# Data Description

**Univariate Stats First Glance**



As we can see, we are unable to explore the data in detail due to various missing values and incompatible data types.

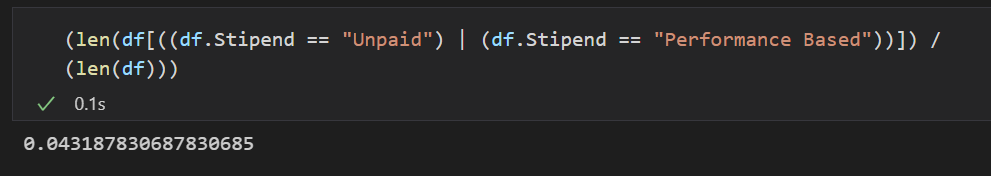
# Data Cleaning

## Stipend

For our case, we decided to begin cleaning the dataset starting with our target variable, i.e., Stipend.

We saw that there are two categories in the column. One is paid and another is unpaid and performance based. But since the aim of our project is to perform analysis and build model around estimating stipend amount, we decided to filter out unpaid stipend internships as it doesn’t serve much of a purpose in this project.

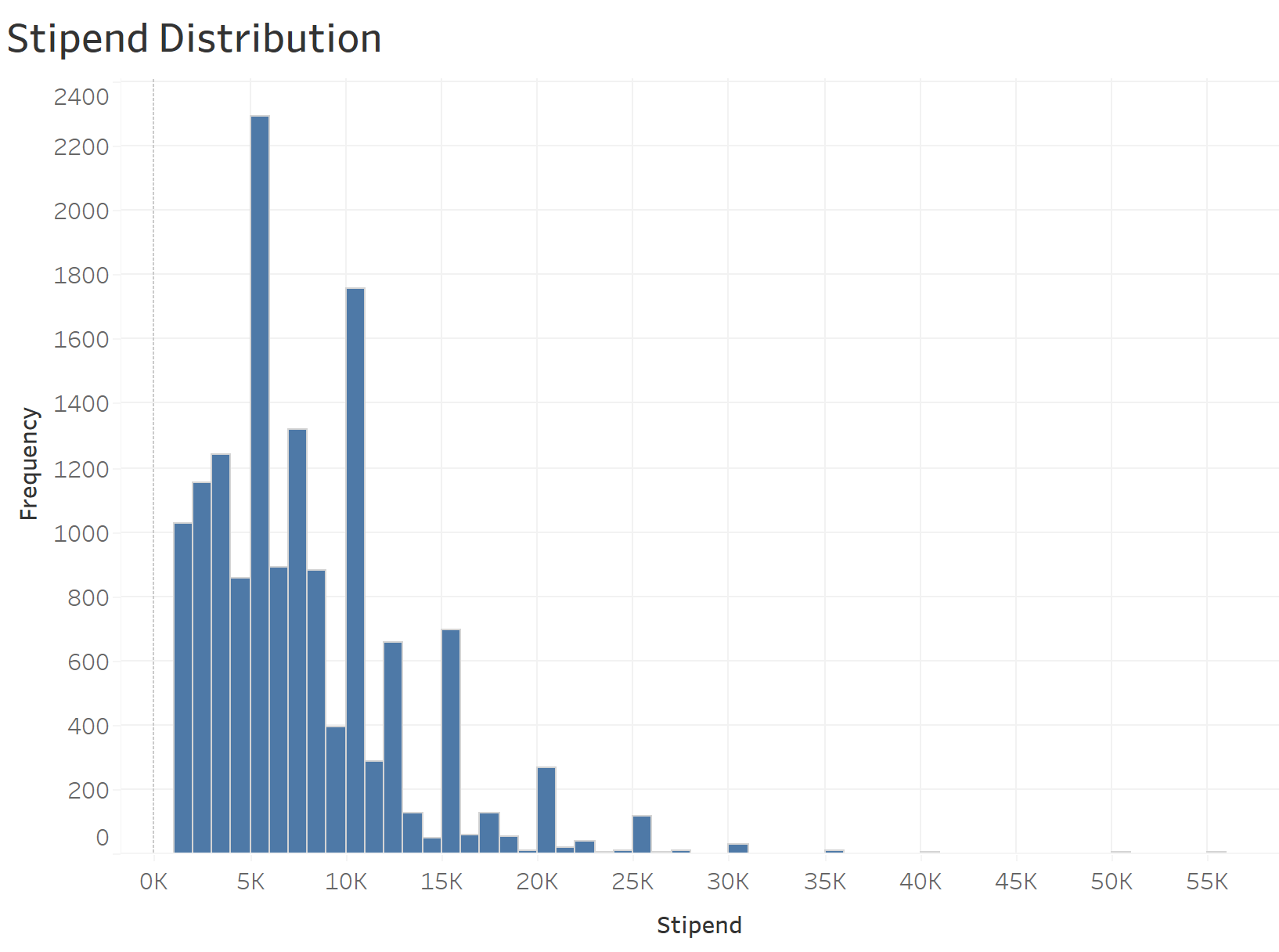
Also, we saw that unpaid internships constituted only 4.3% (653 observations out of 15120 total observations) of our entire dataset.



Next, we proceeded to feature engineer a new feature “Incentives” if it is being provided alongside with stipend.

**(It would be interesting to see if providing incentives as an extra had any affect on the amount of stipend being offered.)**

We also saw that there was a wide range of different types of values in stipend, from range of values to monthly to weekly to per design basis, and also some corrupted data. We cleaned all that up, performed conversions as and when necessary and parsed only the stipend per month numeric amount.

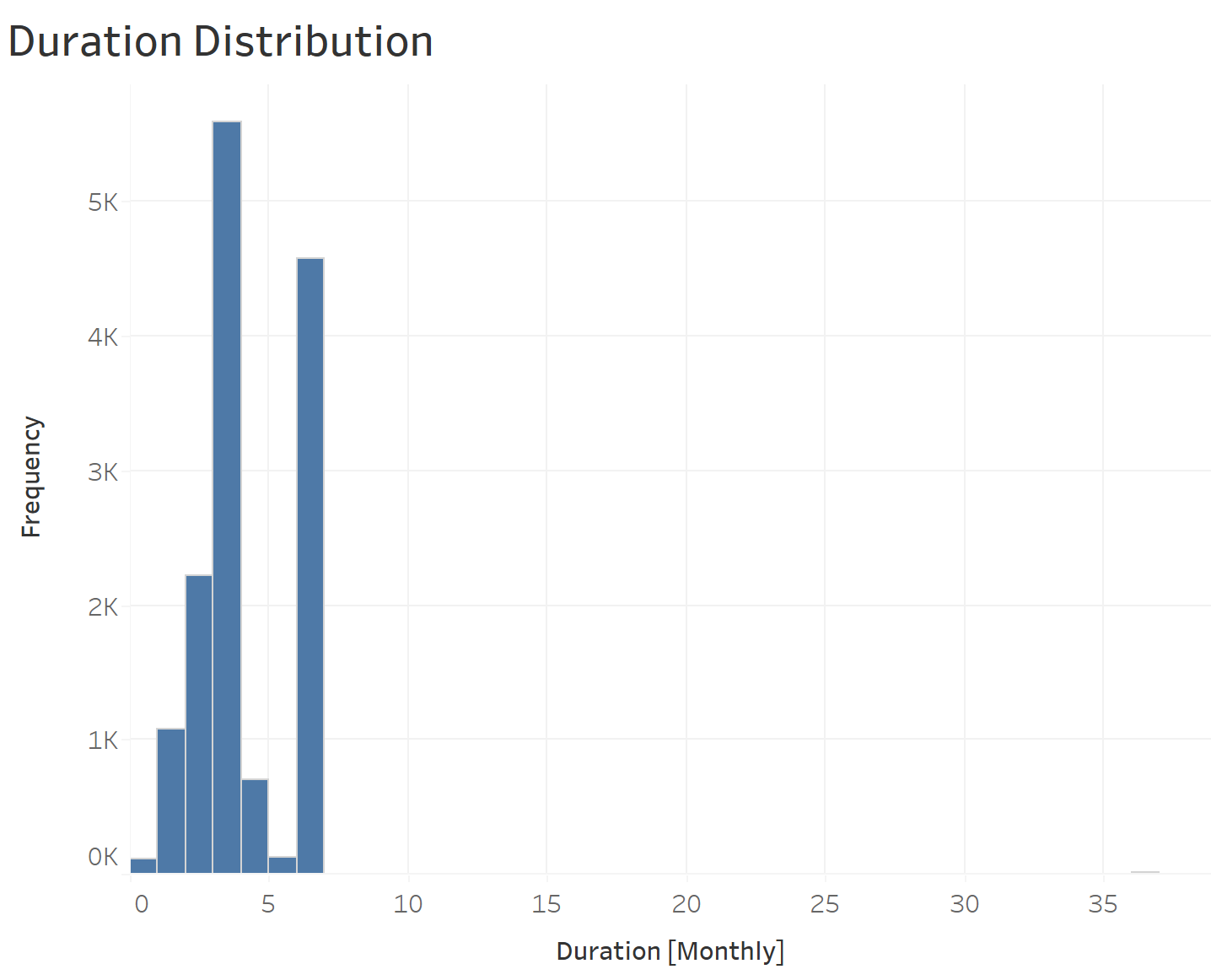


Skewness: 2.09, Right Skewed

Kurtosis: 13.57, Leptokurtic

## Duration

Here, again we saw many anomalies. Some data were in monthly format, some weekly, some on days basis, some corrupted. We cleaned all that up, performed conversions as and when necessary and parsed only the duration in number of months.



Skewness: 3.38, Right Skewed

Kurtosis: 49.82, Leptokurtic

## Location

Here, we see that there are many locations that has very few occurrences (2 or less). We expect that these locations are less popular and should not exhibit a significant difference when predicting stipend.

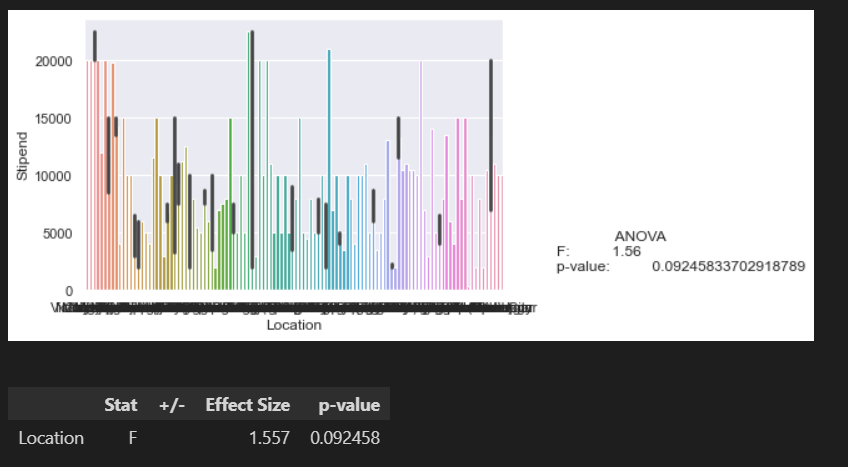
So, we decide to perform a hypothesis test to verify if they are significant when predicting stipend.

***Null Hypothesis (H0) :*** *All locations with 2 or less occurrences do not exhibit a significant difference when predicting stipend.*

***Alternate Hypothesis (HA) :*** *All locations with 2 or less occurrences do exhibit a significant difference when predicting stipend.*

We get the following result after performing One-Way ANOVA test:

**One-Way ANOVA test result**



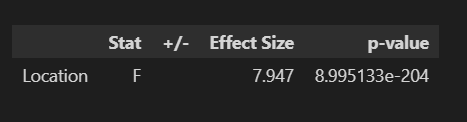
Since, we find that p-value (0.092) > 0.05,

We fail to reject the ***Null Hypothesis (H0)***.

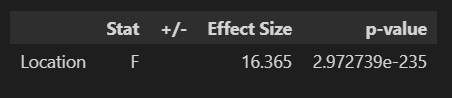
Hence, we conclude that these locations can be grouped under a single category “Other” for our further analysis as they are statistically insignificant.

After grouping these less popular locations as “Other”, we saw a significant improvement of One-Way ANOVA scores when compared.

**One-Way ANOVA result before grouping**



**One-Way ANOVA result after grouping**



## Title

Here, we see that there are many titles that has very few occurrences (2 or less). We expect that these titles should not exhibit a significant difference when predicting stipend.

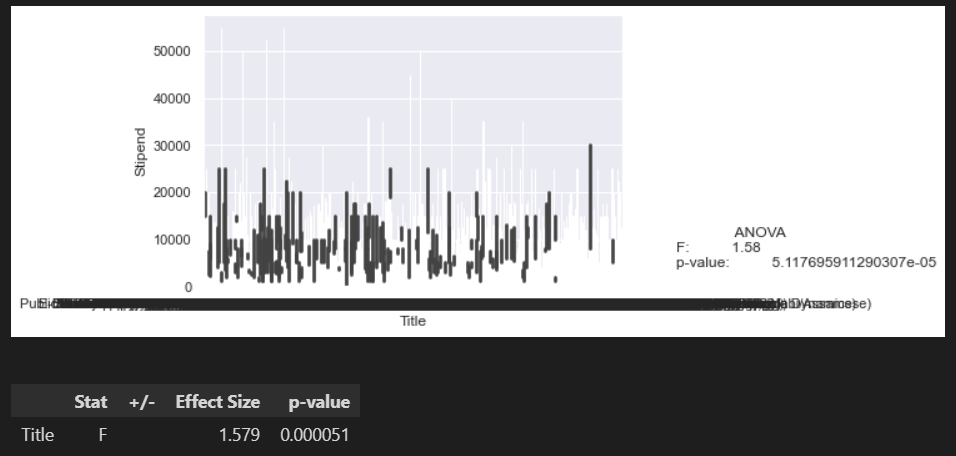
So, we decide to perform a hypothesis test to verify if they are significant when predicting stipend.

***Null Hypothesis (H0) :*** *All titles with 2 or less occurrences do not exhibit a significant difference when predicting stipend.*

***Alternate Hypothesis (HA) :*** *All titles with 2 or less occurrences do exhibit a significant difference when predicting stipend.*

We get the following result after performing One-Way ANOVA test:

**One-Way ANOVA test result**



Since, we find p-value (0.000051) < 0.05,

We reject the ***Null Hypothesis (H0)***.

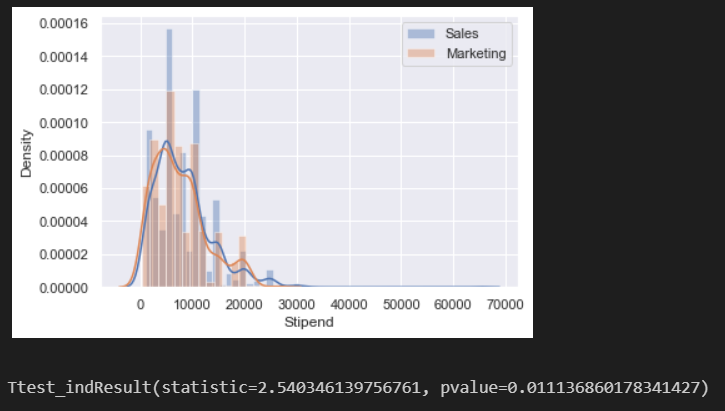
Hence, we conclude that these titles cannot be grouped together as they are statistically significant.

We clean the data further by renaming some same terms with different names.

We also perform a hypothesis with a t-test that "Marketing" and "Sales" in title feature are similar and their effect is not significantly different on the target variable.

We get the following result after performing the t-test:

**Sales and Marketing t-test result**



With a p-value of < 0.05, we conclude that there is a statistically significant difference and we cannot rename them with a similar term.

## Applicants

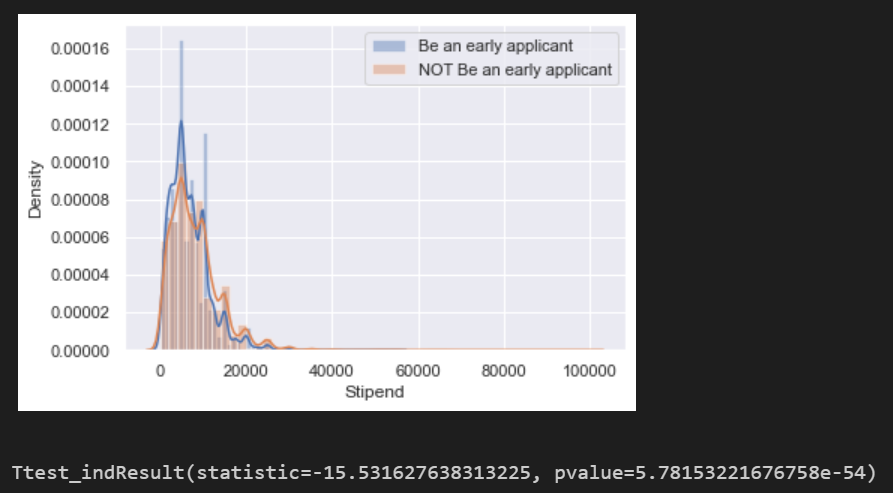
We perform a t-test for a hypothesis that having “Be an early applicant” in applicants feature is no different than not having it.

***Null Hypothesis (H0) :*** *There is a no significant difference between having “Be an early applicant” in applicants feature and not having it.*

***Alternate Hypothesis (HA) :*** *There is a significant difference in having “Be an early applicant” in applicants feature and not having it.*

We get the following result after performing the t-test:

**t-test result**



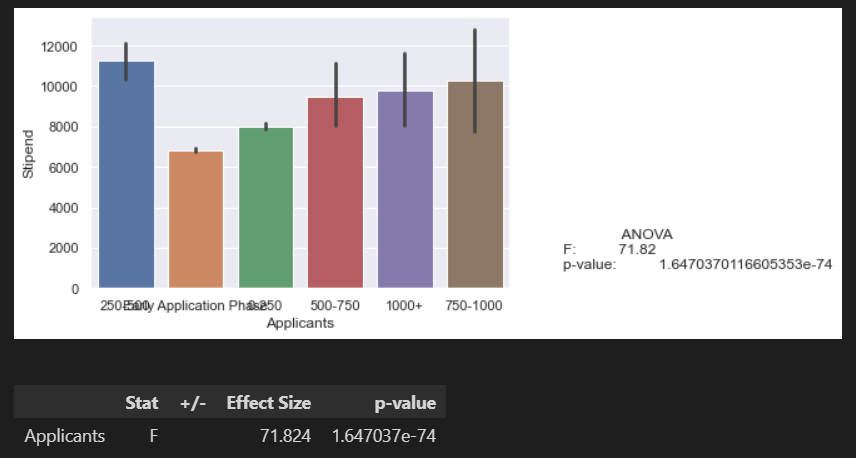
Since, we find p-value < 0.05,

We reject the ***Null Hypothesis (H0)***.

We conclude that there is a statistically significant difference between having and not having “Be an early applicant” in applicants feature.

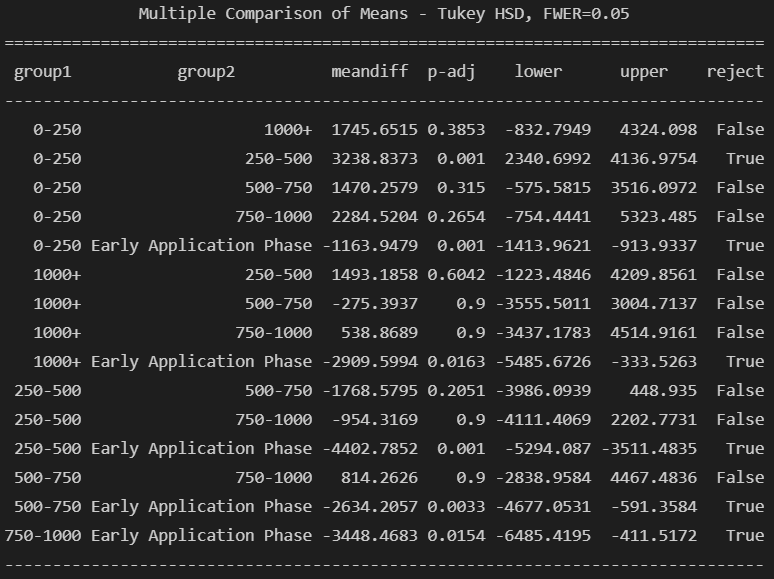
Then we create multiple bins and check for significance with One Way ANOVA test.

**One Way ANOVA f-test result**



We see that the groupings are statistically significant. We also perform a Tukey's HSD (honestly significant difference) test between each pair to confirm the significance.

**Tukey HSD test result**



## Skills Required

For Skills Required feature, we see a lot of missing information along with varieties of skill combinations each separated with a comma.

We replace the missing values with “Not Specific” based on the assumption that there are no specific requirements of particular skills.

Then we parse out the Skills separated by commas into lists to create dummy variables later with Multi Label Binarizer for each observation.

(Use visuals here)

## Perks

Here, again we see a missing information along with varieties of perks combinations each separated with a comma.

We replace the missing values with “NA” based on the assumption that no perks are being provided.

Then we parse out the Perks separated by commas into lists to create dummy variables later with Multi Label Binarizer for each observation.

(Use visuals here)

## Number of Openings

We don’t see many discrepancies here other than 4 missing values. We fill them with the mean and proceed further.

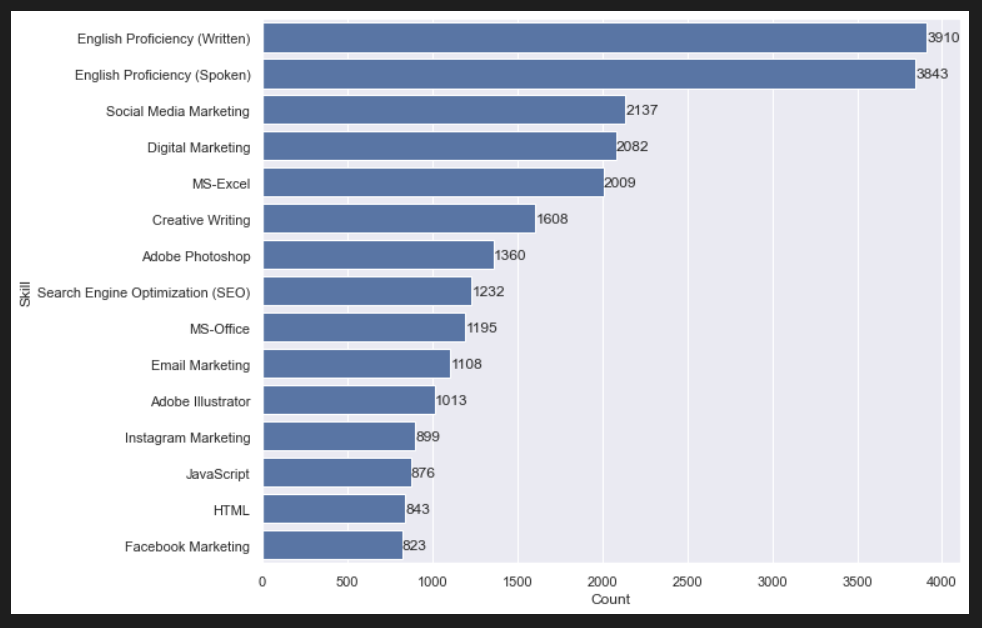
# Data Exploration

We explore our data from various angles to find all interesting points of interest.

## Skills Required

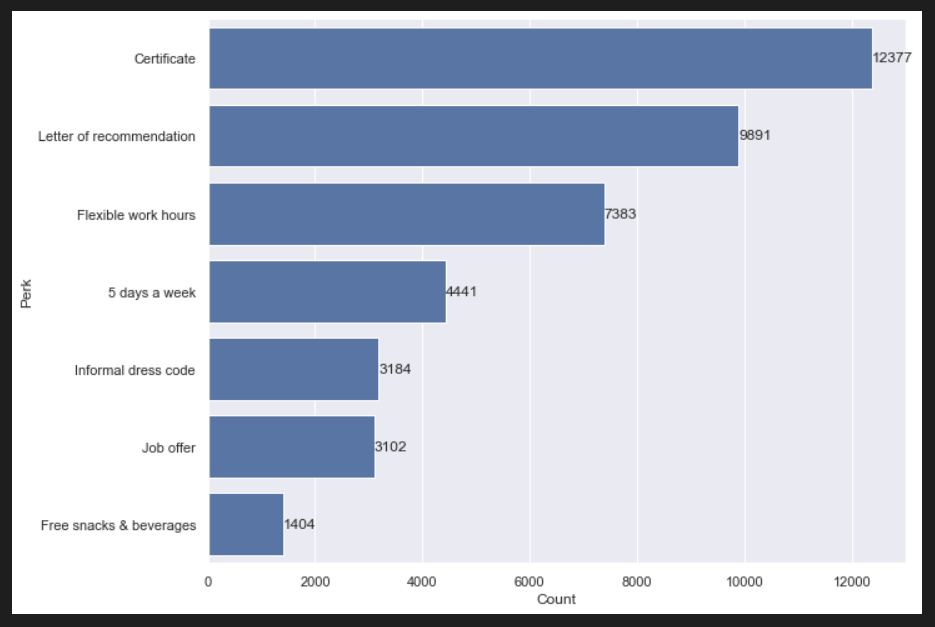
We explore the top 15 most required skills for internship positions.

**Top 15 most popular skills**



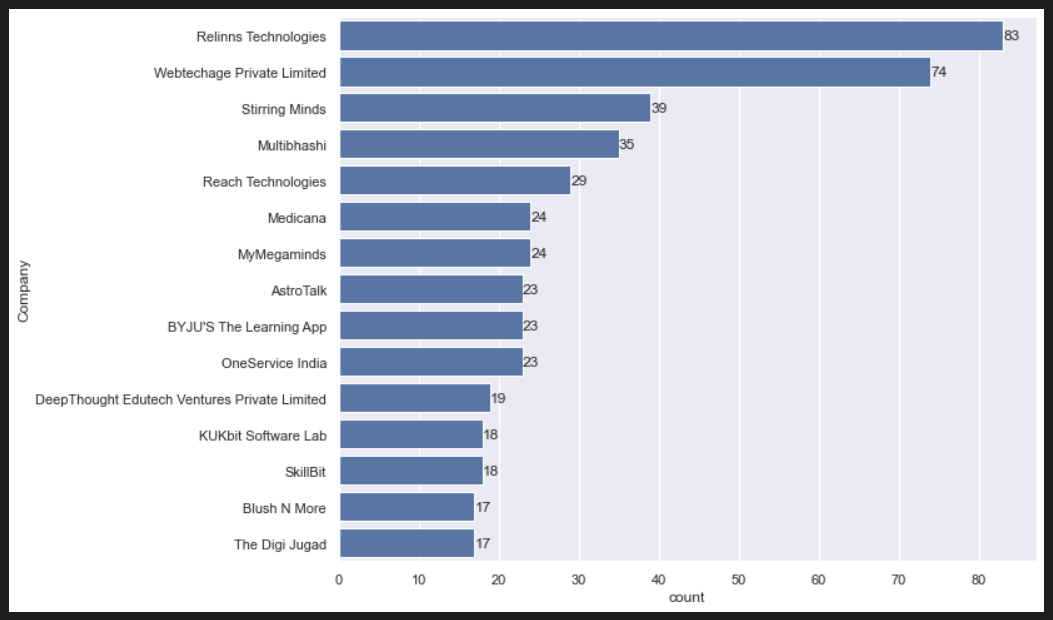
## Perks

We explore all the most offered perks in internship positions.



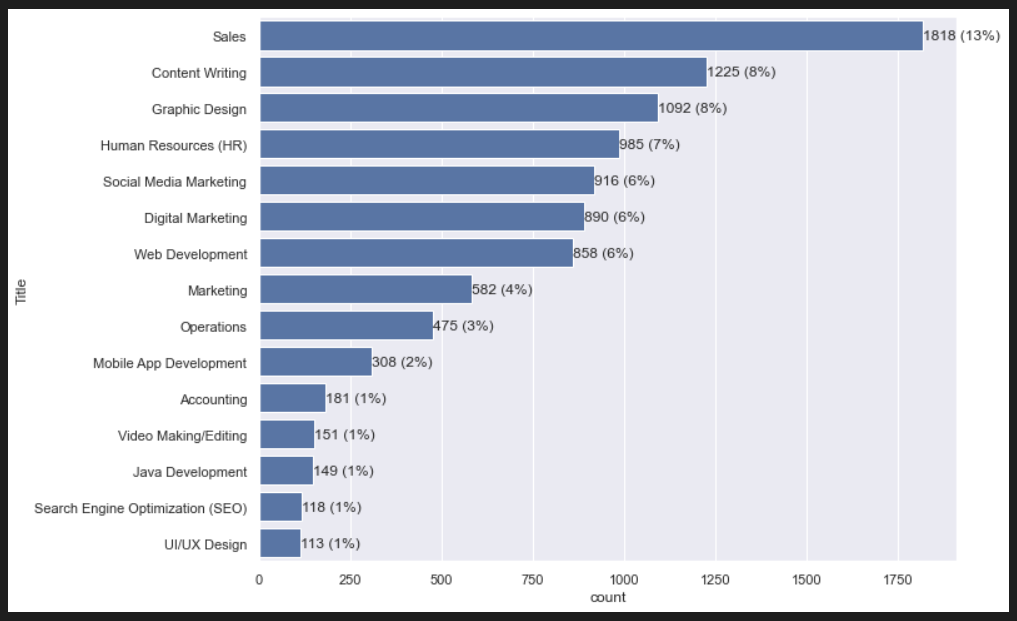
## Company

We explore the top 15 hiring companies.



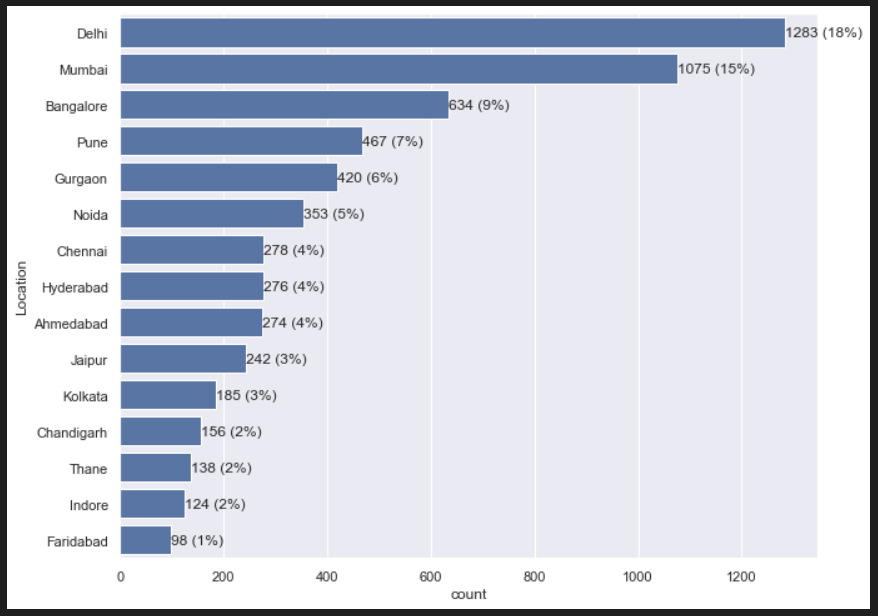
## Title

We explore the top 15 titles required for internships.

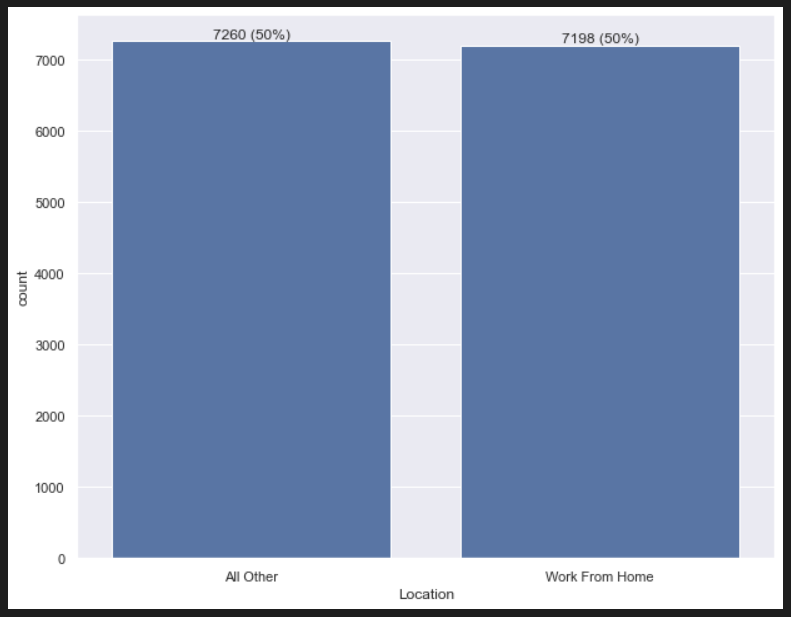


## Location

We explore the top 15 locations offering most in-office internships.



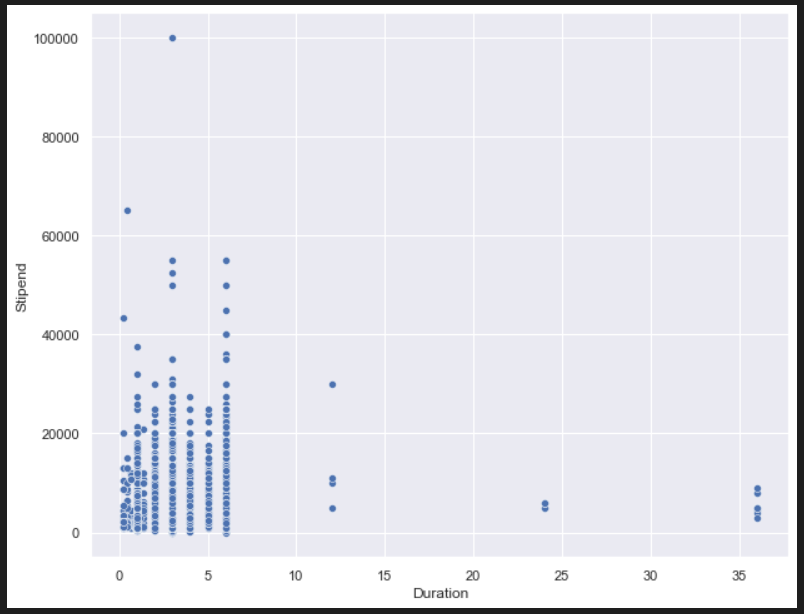
Next, we also explore the ratio of in-office internships to Work from Home internships.



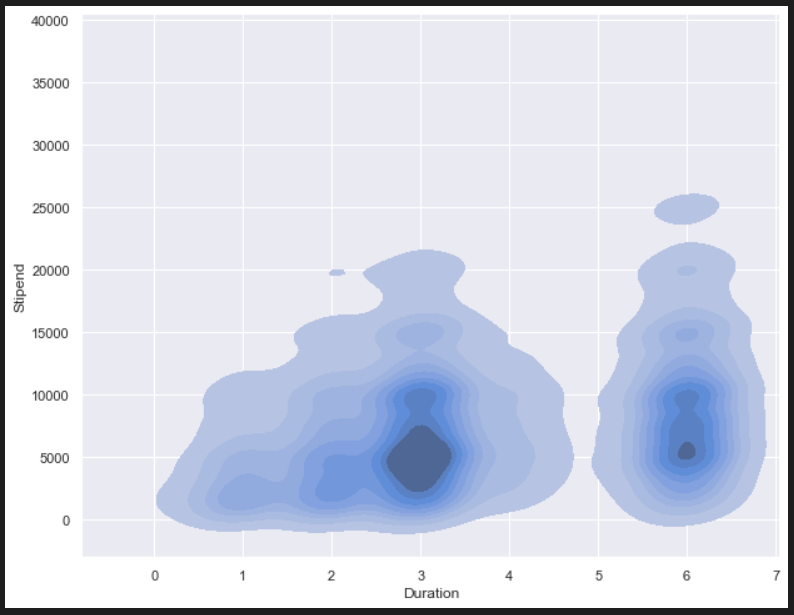
We see that the ratio is almost 1:1.

## Duration – Stipend Relationship

We explore the relationship between Duration and our Target Variable Stipend with a scatter plot.



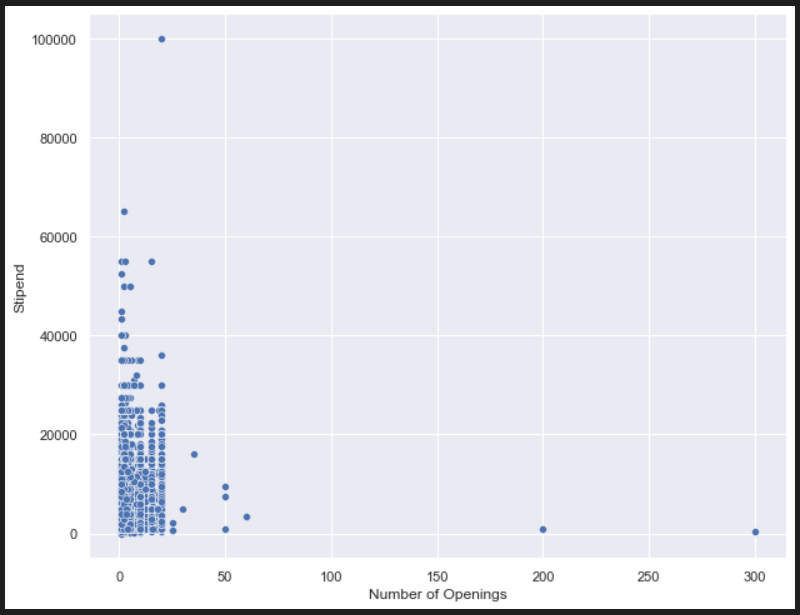
We see that the plot is ineligible to read as the data is stacked together so we remove the outliers and plot a kernel density estimate (KDE) plot for better visualisation.



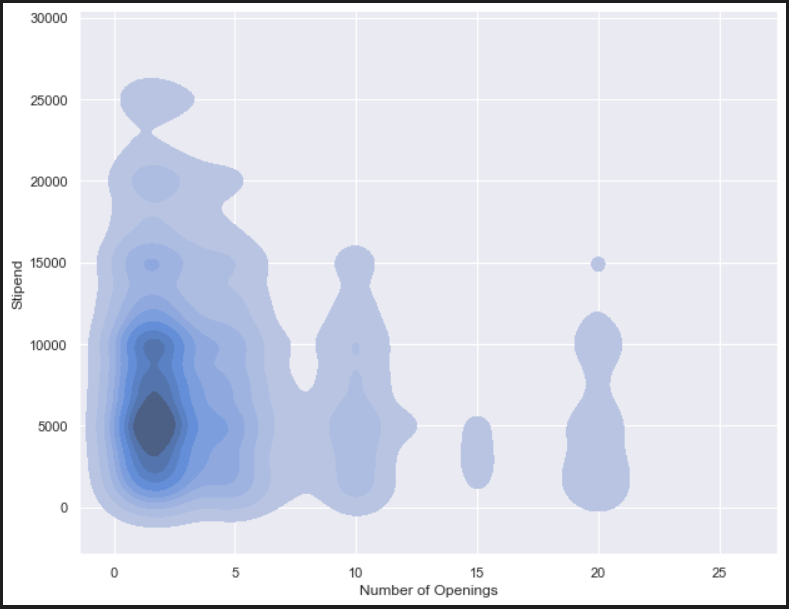
Here, we can see that majority of our data is centred in and around Duration of 3 months and Stipend of 5000. Some density can also be seen centred at around Duration of 6 months and Stipend of 5000.

## Number of Openings – Stipend Relationship

We explore the relationship between Number of Openings and our Target Variable Stipend with a scatter plot.



Again, we see that the plot is ineligible to read as the data is stacked together so we remove the outliers and plot a kernel density estimate (KDE) plot for better visualisation.



Here, we can see that majority of our data is centred in and around Number of Openings of 0 to 5 and Stipend of 5000.

# Target Variable Transformation

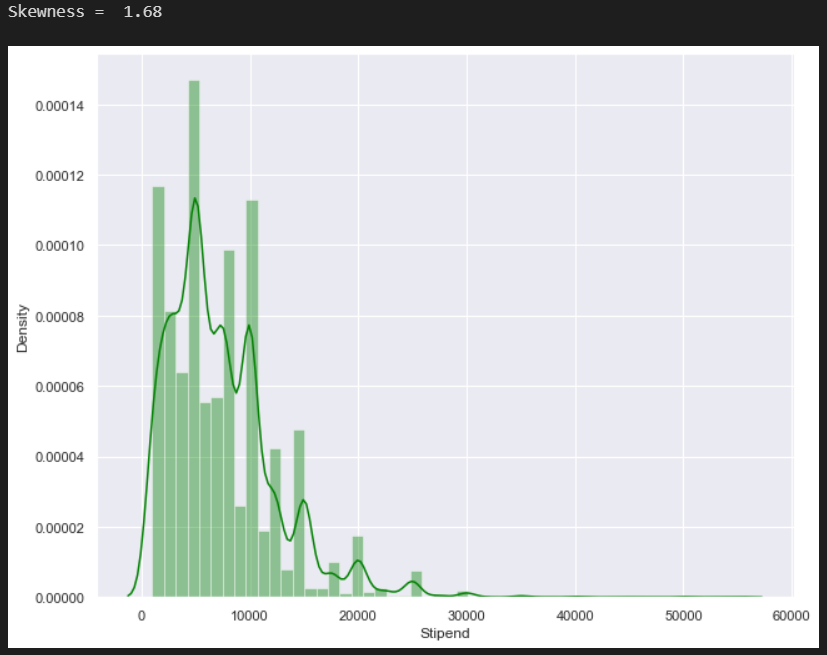
We see that our target variable Stipend has the following properties:

*Skewness: 2.09, Right Skewed*

*Kurtosis: 13.57, Leptokurtic*

We first remove the outliers of Stipend greater than 60000 and less than 1000 as seen in the scatter plot in above chapter.

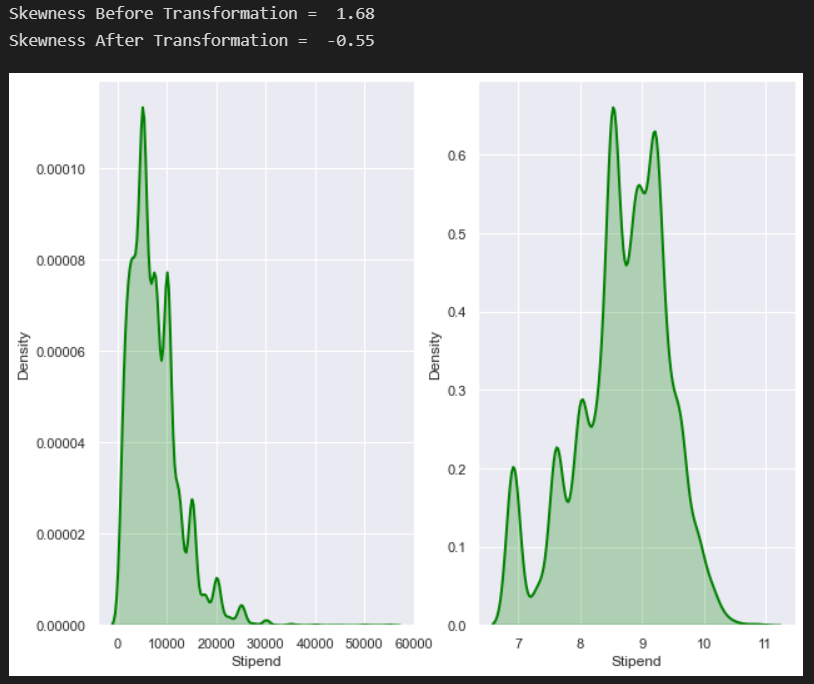
Then we plot a histogram and check the skewness again. We see that it has reduced to 1.68.



We perform the following transformations to try to normalise the Target Variable:

## Log Transformation

We use the log method from NumPy to perform log transformation on our target variable and then check for skewness.

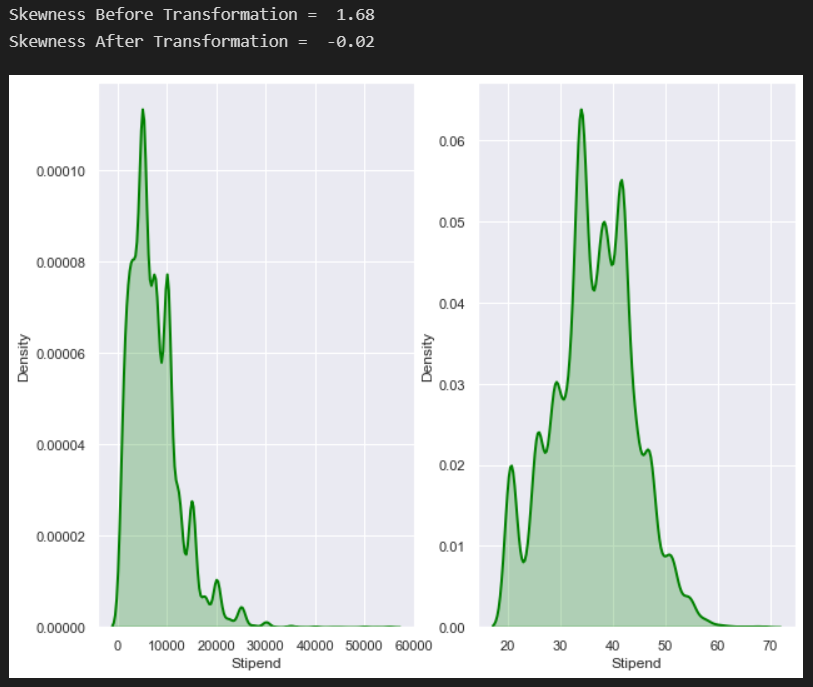


We see that it has reduced to -0.55.

## Box-Cox Transformation

First, we find the optimal lambda value for our Box-Cox transformation which provides the best possible skewness.

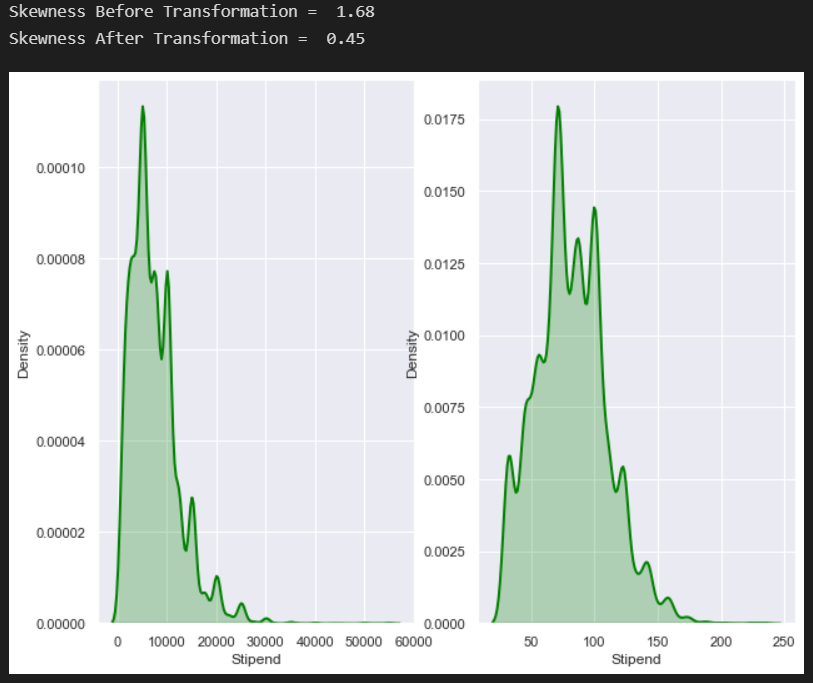
Then, we use the boxcox method from SciPy to perform Box-Cox transformation on our target variable.



We see that our skewness is minimal at -0.02.

## Square Root Transformation

Here, we use the sqrt method from NumPy to perform Square Root transformation on our target variable and then check for skewness.



We see that our skewness is at 0.45.

We store all the transformations in our DataFrame and proceed forward. We will choose the one that give the best scores.

# Creating Dummy Variables

We use the MultiLabelBinarizer from sklearn to create dummy variables of all skills and perks for each observation.

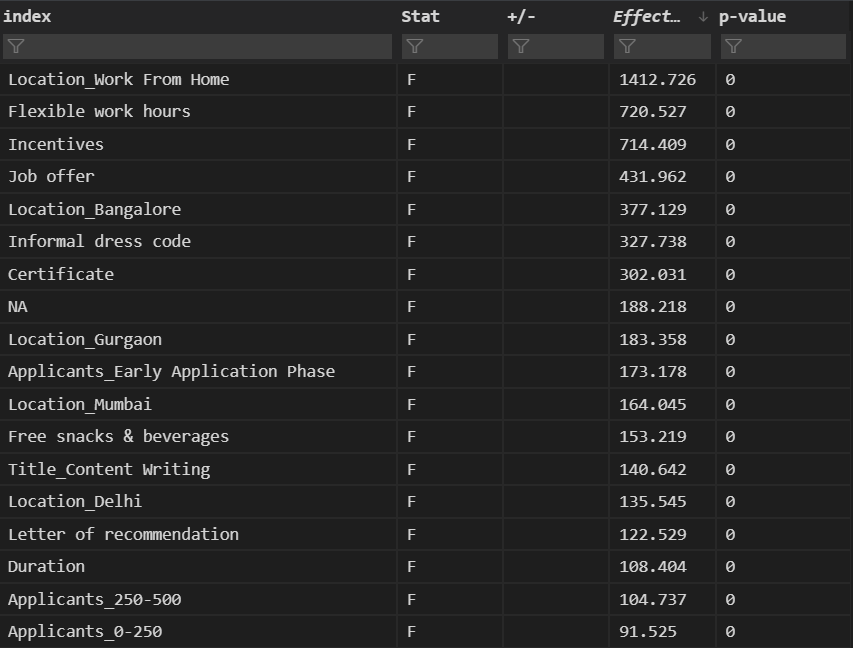
Then after creating dummies for all other categorical features such as Title, Location and Applicants, we concatenate everything together in one DataFrame.

# Pre-Modelling Tests

## One-Way ANOVA

We perform One-Way ANOVA test on all features with our Target Variable to check for all individual statistical significances.

We see that some of the most significant features are as follows:

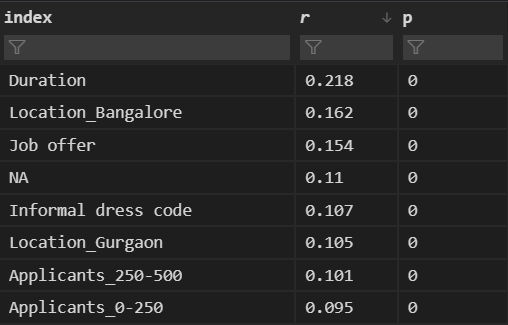


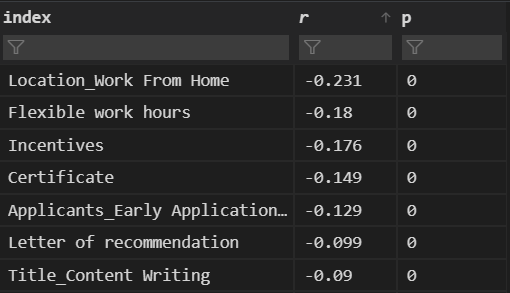
Although, we see there are many statistically insignificant features, we don’t decide to drop them for now as we can reduce the complexity during our model building process with regularization.

## Pearson Correlation

We calculate and check the Pearson Correlation Coefficients for all features with our target variable with their respective p-values.

We see that some of the most correlated features are as follows:





## Variance Inflation Factor (VIF)

We skip performing Variance Inflation Factor (VIF) test in this case as there is no suspicion of multicollinearity in our model.

# Model Building

## Selection of Model

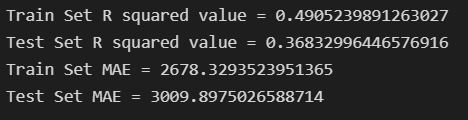
For our Dataset, we decide to use Light Gradient Boosting Machine (LGBM) Regressor.

The reason being is that our Dataset is quite large (14,000+ observations) and Light GBM is very good at handling large size of data (above 10,000) while also staying efficient as it takes lower memory to run. It also performs better on Datasets with higher complexity as it provides options for regularization and can also be further hyper-parameter tuned to improve the balance between bias and variance.

## Selection of Transformed Variable

We fit the features with all our transformed target variables and also with the non-transformed target variable on the vanilla LGBM model and evaluate the scores.

Then, decide to proceed with Box-Cox transformed Target Variable for our modelling process as that provided the best baseline scores.



## Hyper-Parameter tuning of the Model

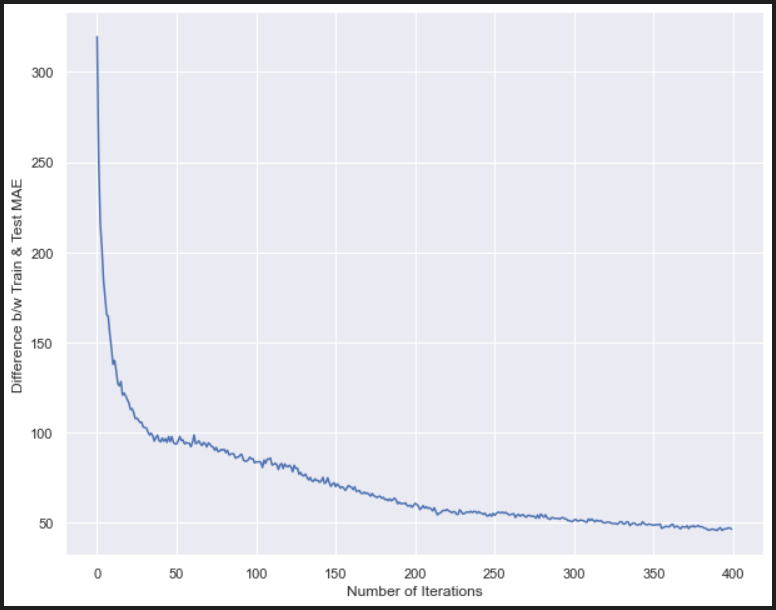
### Regularization

Our primary objective of Hyper-parameter tuning in our case is to reduce the complexity of our model. This is because we have a large number of features (2499) in our model which can cause overfitting.

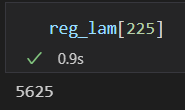
For this purpose, we use the “reg\_lambda” hyper-parameter in our LGBM model to provide L2 Regularization. We use L2 Regularization because it disperses the error terms in all the weights which can lead to more accurate final model that can generalize better.

To find the ideal value of Lambda of L2 Regularization, we run a For-Loop to plot the difference between the Mean Absolute Error of the training set and the test set.

We get the following result:



We see that the difference between the MAE of the training set and the test set gets minimal from around 225 iterations so we decide to take the lambda value from that point which is 5625.

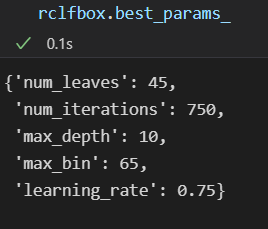


### Randomized Search CV

We perform a randomized search with the objective to find the best hyper-parameters for our model.

We set our scoring method to Mean Absolute Error to find the best hyper-parameters on. We decide to choose MAE for evaluation because it is relatively easy to interpret and outliers are not particularly bad in for this type of model.

After completion of Randomized Search CV, we find the following best hyper-parameters for our model:



Our MAE Scores are as follows:



### Feature Importance

We see that some of the most important features in our model are as follows:



# Saving of the Model

We save the model in a pickle file along with all other necessary objects that we plan to use further in our Flask API to fetch the Stipend predictions as and when requested by user.

# Productionizing the Model

We finish the project by productionizing it by building a Flask API and hosting it online that takes in inputs from the user such as location, skills, perks, duration, etc. and returns an estimated stipend amount.

[Link of Productionized Model](https://flaskinternshalamodel-production.up.railway.app/)