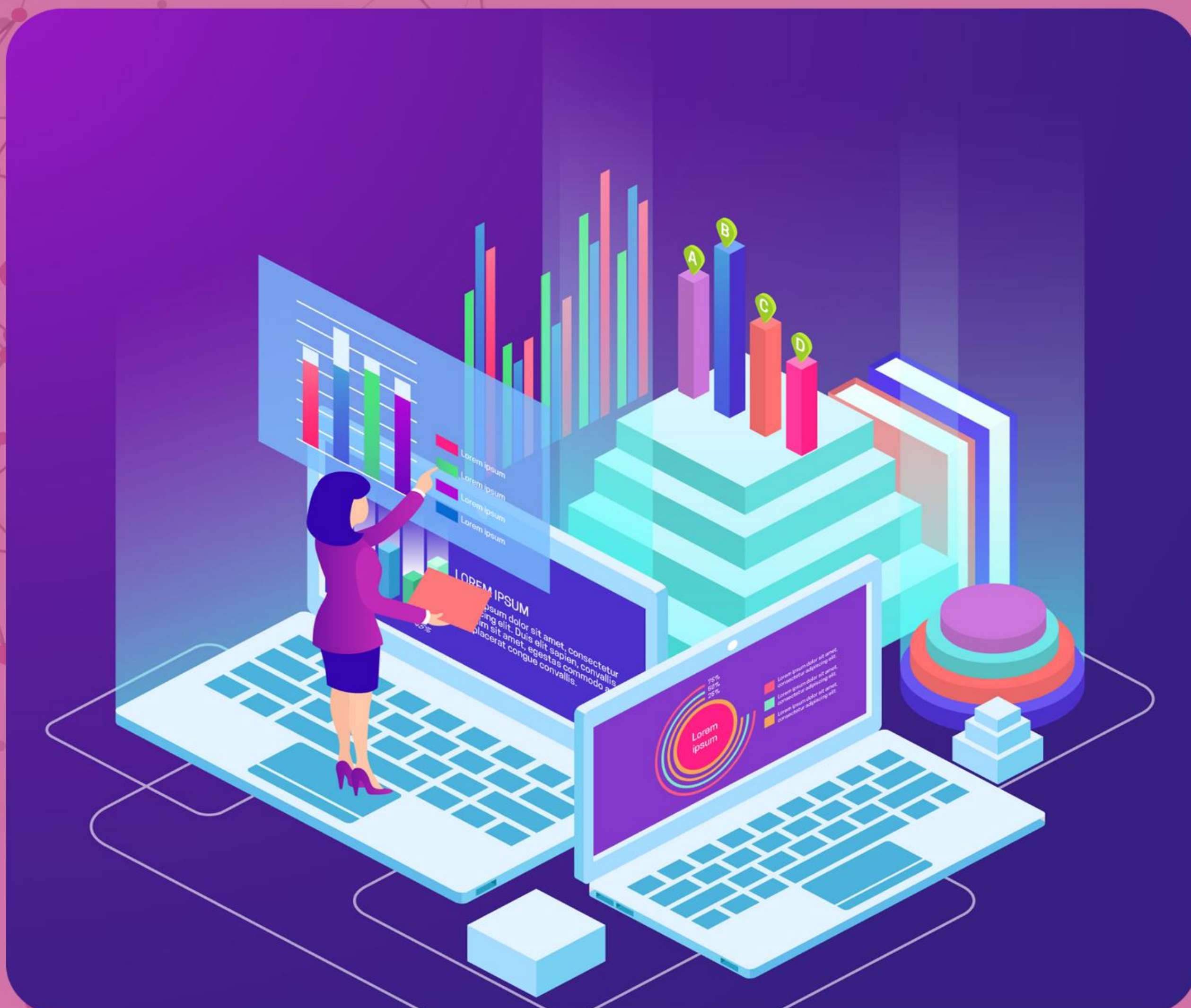
SALARY PREDICTION FOR DATA SCIENCE CAREERS



GROUP 05

SANUJI DEVASURENDRA - S14810
SAHAN MADHUSHANKA - S14835
ISINI ASALYA - S15023

Abstract

This study brings out both a descriptive analysis and an in-depth analysis of the salary prediction for data science-related careers. The main objective of this analysis is to provide not only employees but also any other party who needs access to salary information related to data science careers around the world. Therefore, for this forecasting, initially an exploratory analysis was performed. Thereafter, after fitting several statistical models based on root mean square error (RMSE) and mean percentage error (MAPE) judgments, it was identified that the best model is gradient boosting regression to explain this salary prediction model.

Table of Contents

List of figures

List of Tables

1. Introduction

Data science has emerged as one of the most lucrative and rapidly growing fields in recent times. The demand for jobs related to data science is at an all-time high, as organizations across various industries are looking to leverage the power of data to make more informed decisions and gain a competitive edge. As a result, the salary prospects for data scientists have never been better. Hence, through this analysis, we will first explore the various factors that influence data science salaries and provide insights into the average salaries for different data science related occupations. Thereafter, by fitting several statistical models and using the optimal model, we will predict the optimal salary for an applicant under the factors considered in the analysis.

2. Description of the Question (include objectives)

Several educational institutions are enticed to develop academic programs linked to artificial intelligence, neural networks, big data, and other similar topics since individuals working in the field of data science earn excellent incomes. As a result, this aspect of "high-end pay" seems to influence not only employees but also young students.

Hence, our study focuses on "**predicting the salary (in USD) of an employee in the field of data science.**" Furthermore, the goal of this project is not only to provide the best estimate of salary for data science professionals or young, passionate students to get some motivation for where they should be financially in several years with some notion about their future pay, but also for organizations to have a sense of the current salary trend for different job positions in the market for better budget planning in salary increments, human resource compensation, and benefits.

To accomplish the above objective, sub-objectives of this project are to,

1. Perform an exploratory data analysis to identify the most influential predictors of the response.

2. Develop a predictive model that incorporates the most significant factors to forecast the predictor.

3. Description of the Dataset

The dataset "Salaries" is sourced from https://ai-jobs.net/salaries/download/ and it is a collection of 2904 observations with no missing values. The dataset consists of 11 variables and their description is as follows.

No.	Variable Name	Description
1	wrok_year	The year during which the salary was paid. Categories: 2020,2021,2022,2023

2	experience_level	Categories: EN = Entry Level; MI = Mid-Level; SE = Senior-Level; EX = Executive-Level
3	employment_type	Categories: PT = Part-time; FT = Full-time; CT = Contract; FL = Freelance
4	job_title	The job role through the year. There are 83 categories.
5	salary	The total gross salary amount paid (p.a.)
6	salary_currency	The currency of the salary (p.a.) paid as an ISO 4217 currency code.
7	salary_in_usd	The salary in USD per annum (FX rate divided by the average USD rate for the respective year via fxdata.foorilla.com)
8	employee_residence	Employee's primary country of residence during the work year as an ISO 3166 country code
9	remote_ratio	The overall amount of work done remotely, possible values are as follows: 0 = No remote work (less than 20%); 50 = Partially remote; 100 = Fully remote (more than 80%)
10	company_location	The country of the employer's main office or contracting branch as an ISO 3166 country code
11	company_size	The average number of people that worked for the company during the year: S = less than 50 employees (small); M = 50 to 250 employees (medium); L = more than 250 employees (large)

References:
Country codes: https://www.nationsonline.org/oneworld/country_code_list.htm Currency codes: https://www.iban.com/currency-codes

3.1. Data Pre-processing

In the data pre-processing, first salary variable and salary currency variable are removed because the objective is to predict the salary in USD. Thereafter, the company's location and the employee’s residence were categorized based on continents rather than an analysis based on individual countries. Next, we filtered only the full time employees in favor of the objective, where we disregarded part-time, contract, and freelance employees.

Additionally, we created a new variable called Role Type that categorizes the 83 job titles into the following nine major categories, considering their duties.

Data Strategist	Data Architect	Data Analyst	Data Scientist	Data Engineer

Business Intelligence Analyst	ML Ops Engineer	Data Product Manager	Other

Table 3.1.1 Classification of Data Science Job Roles

Further, a new binary variable called "leadership" was created based on their leadership role to assess whether there is any significant impact on the response.

4. Important results of the descriptive analysis

As for the objective of developing a predictive model, it is essential to understand the relationship between response variable and predictors.

Therefore, starting with the response, which is “Salary in USD” per annum, from Figure 4.1 and Table 4.1, the mean salary is slightly greater than the median. However, since the skewness statistic is less than one, it implies that our distribution is approximately normal.

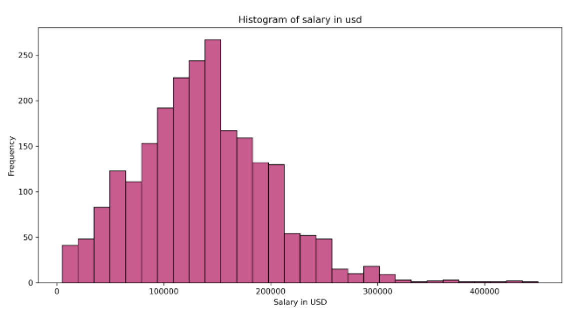


Figure 4.1 Histogram of salary in USD

<u>Salary in USD (p.a) statistics</u>

Minimum: 5,132
1st quartile: 95,000
Median: 134,000
3rd quartile: 172,200
Maximum: 450,000
Mean: 136,011.18
Skewness: 0.554606

Table 4.1 Statistics of Salary in USD

Further, it was discovered through the ground analysis that some organizations, including META, Microsoft, and Amazon, will pay very high salaries for senior employees, such as principal data scientists and senior data scientists, who aim for maximum annual salaries of $550,000 according to the Glassdoor records. Due to the presence of the aforementioned employees in our dataset, the extreme values in our data cannot be regarded as outliers.

Moving on to the analysis of predictors with the salary in USD, first salaries with respect to the work years were analyzed.

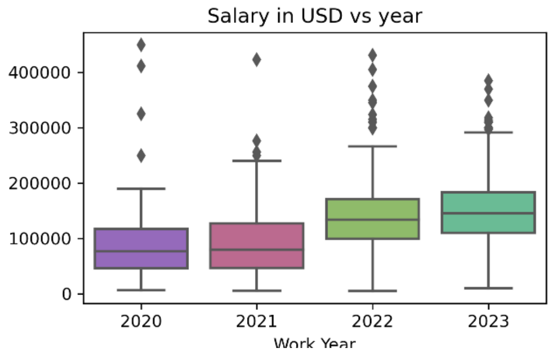


Figure 4.2 - Salary in USD vs. Work Year

It can be observed that, when compared to the years 2020 and 2021, the most recent two years contain a price hike in Figure 4.2. According to the *Burtch Works Study DS Analytics (2021),* the possible reasons for this fluctuation are increased demand and popularity of data science, advancements in technology, and economic conditions. Further, if an explanation of the economic condition is provided, since with recent inflation, many organizations around the world have introduced the pegged salary system. Hence, as a result, salaries started to grow in different countries.

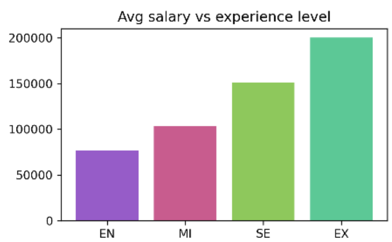


Figure 4.3 Average salary across experience levels.

Next, to explain the salary variation across the experience level, it is generally known that the higher you walk through the organization's hierarchy, the higher you get paid. This variation can be clearly explained by Figure 4.3, in which Glassdoor too makes a similar conclusion the higher the experience, the higher the pay.

Further, as in Figure 4.4, when we consider the salary variation across seniority over years, it too highlights that the salaries across each level approximately increase for each year.

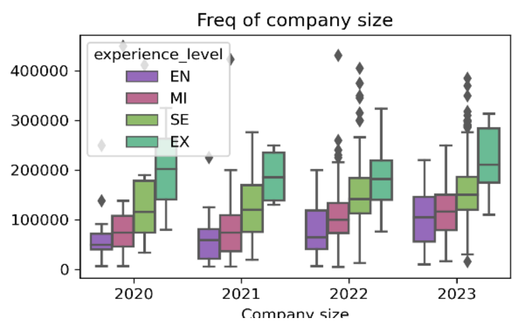


Figure 4.4 Salary variation across experience levels over the years

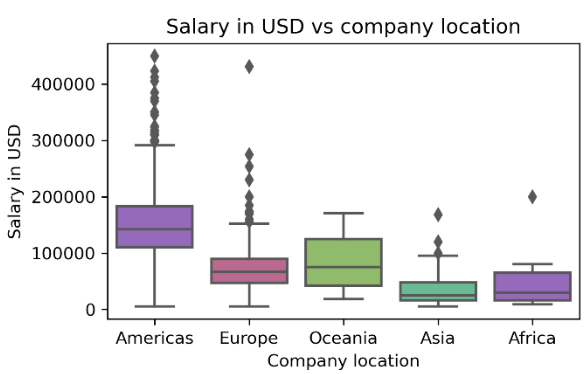


Figure 4.5: Salary variation across the company location

Next, to move on with the salary variation across the company locations, the company locations were classified based on the continents. According to Figure 4.5, companies in the United States, Europe, and Oceania pay more than Asian and African companies.

The existence of major organizations like Apple, Microsoft, and Amazon across developed continents may be a reason for well paying. Additionally, the fact that Africa is the least developed continent according to the United Nations Conference on Trade and Development may be the cause of its low employment wages.

Further, because the majority of Asian nations have developing economies, pay averages can be lower in Asian companies than in the West, as shown by Figure 4.5.

Moving on to compare the salaries across employee residences, according to Figure 4.6, Americans, Europeans, and Oceanians get highly paid. The reason behind the increased salaries on those continents may be because of higher living costs compared to other parts of the world.

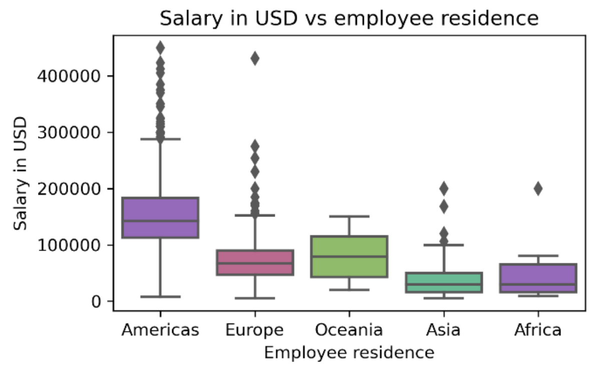


Figure 4.6: Salary variation across the employee residence

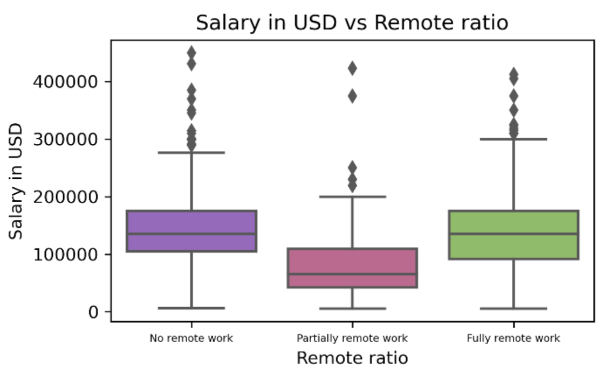


Figure 4.7 - Salary variation across the working mode

Then, to explain the salary variation across working modes, there is no special effect on the salary from the variable; the mode of work (work from home, hybrid, or no remote work). But, in Figure 4.7, it can be seen that there is a reduction in salary for the partially remote workers, while the median salaries of fully remote and no remote workers share the almost same median salary.

In Figure 4.8, it can be seen that the proportion of hybrid workers is very low compared to the other two work modes. Therefore, it can be observed that the lower number of observations has caused the salary variation.

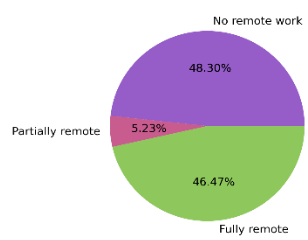


Figure 4.8 Pie chart for working mode

According to Figure 4.9, the majority are from medium sized organizations.

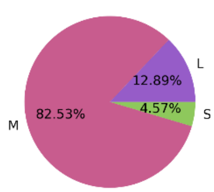


Figure 4.9 Pie chart of company size

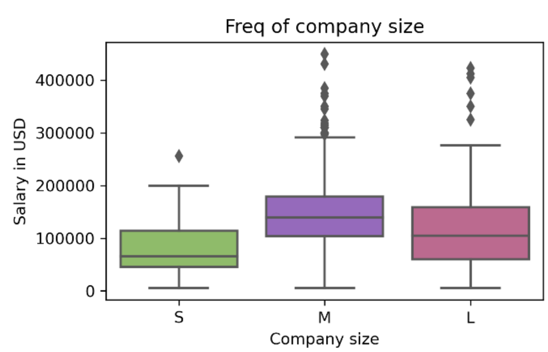


Figure 4.10 Salaries based on organization size

According to Glassdoor records, larger companies pay data scientists 19.5% more than smaller ones, and by observing Figure 4.10, it can be observed that for smaller businesses, the pay is lower than for larger ones. But overall, boxplots suggest that medium companies pay better. A probable reason for this variation is that since the number of employees is limited, companies tend to maximize their pay.

Figure 4.9 illustrates that the subfield of data scientist has extreme pay rates for data scientists. To explain this variation, the existence of highly compensated employment roles like principal data scientists and lead data scientists is most likely one of the causes. However, based on median salaries, data architects are generally well paid. Additionally, jobs like intelligence analysts and data analysts pay less. To further explain this, mastersindatascience.org research states that compared to some of the highly paid career titles like data scientist, data analyst will have fewer skill requirements and a less complicated job description. That could be a reason for lower pay for data analysts.

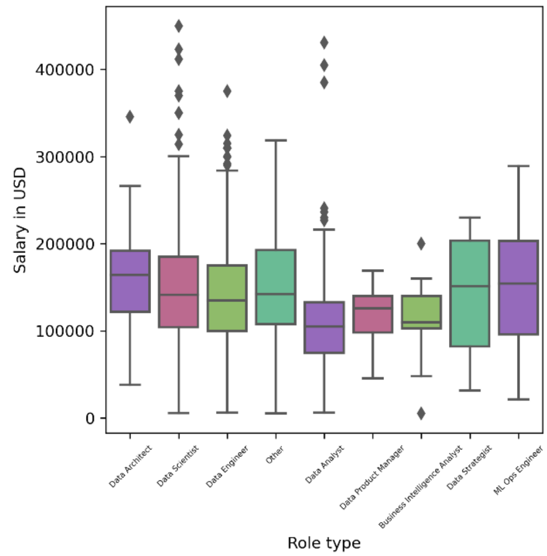


Figure 4.11 Salaries across different subfields in data science

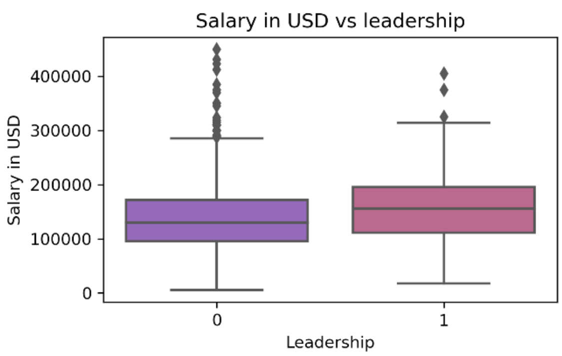


Figure 4.12: Salary variation for leadership roles and others

Lastly, the variation of salaries across leadership roles and non leadership roles is considered. Obviously, when it comes to leadership roles that require more skills and have more responsibilities, salaries are relatively higher compared to others. But according to Figure 4.12, although the median salary is high for leadership roles, there are some non leadership roles that get paid more than leadership roles.

According to figure 4.13, only 6.23% of observations are leadership roles ,while the non-leadership roles are 93.77%. Since the majority are non-leadership roles, that could be a probable reason for this variation.

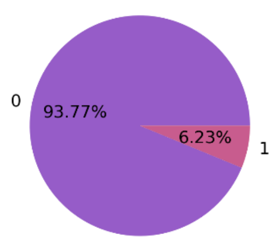


Figure 4.13: Frequency of leadership roles (1) and non leadership roles (0)

5. Suggestions for the advanced analysis

Since "salary_in_usd," the response variable, is a continuous variable, the dataset can be fitted with regression models. Further, the outlier analysis suggested that there are no outliers in the data, which implied that no transformation was necessary because there were not a significant number of outliers.

Hence, moving on to PLSR, partial least squares regression (PLSR) can be used to identify the clusters in the dataset, and grouping the observations into homogeneous groups can be used to improve the accuracy of the fitted models.

However, according to the PLS score plot (Figure 5.1), there are no clusters in the data.

Furthermore, the loadings plot of PLSR can be used to identify the variable clusters and the correlations among the variables.

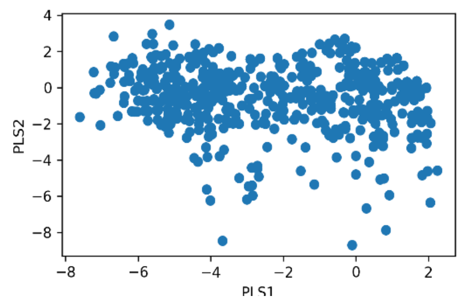


Figure 5.1: Scores plot using PLS

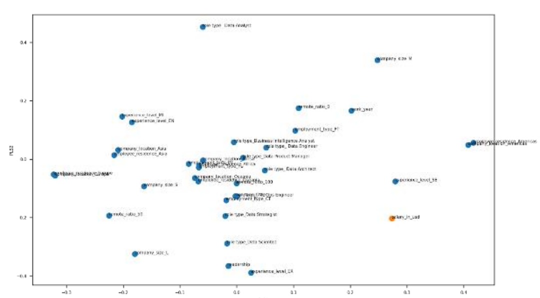


Figure 5.2: Loadings plot using PLSR

The smaller the angle between two variables, the higher the correlation between them.

It is evident from Figure 5.2 that there are some variables that can be grouped together as well as some variables that have a strong correlation with the response variable. (eg: the variable leadership has a higher correlation with salary in USD.)

Moreover, according to the VIF scores in the following table, there is no multicollinearity between the predictor variables since almost all the VIF scores are around 1.

Work Year	Experience level	Remote ratio	Leadership	Company size
1.151204	1.079964	1.055934	1.043131	1.037777

Table 5.1 VIF Scores

6. Important results of the Advanced Analysis

Since the study is focused on a continuous response (salary in USD p.a.), regression models were applied to identify the most appropriate statistical model for the dataset.

6.1. Multiple Linear Regression Model

Starting with the simplest form of fitting data into multiple predictors, a multiple linear regression model is fitted to the dataset. Then the obtained RMSE value is as follows:

RMSE value	48807.0593

Table 6.1.1 RMSE Value - Multiple Linear Regression

But, when a further analysis was carried out to check the multiple linear regression model assumptions, it was identified that the assumption of homogeneity of the variance was violated, as a cone shape variation of residuals can be observed, as illustrated by Figure 6.1.1.

Therefore, shrinkage model techniques were assessed afterwards.

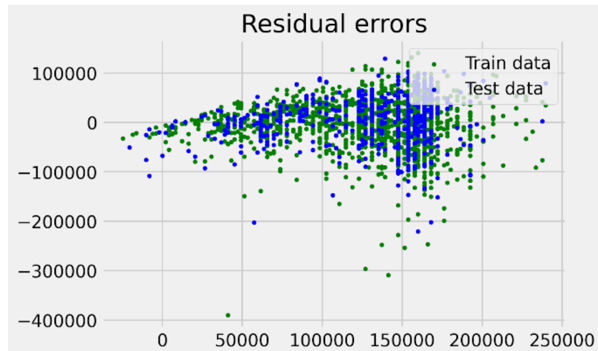


Figure 6.1.1 Residual Error Plot - Multiple LinearRegression

6.2. Shrinkage Technique Models

Considering Ridge, Lasso and Elastic Net regression models, the following outputs were generated.

	Ridge	Lasso	Elastic Net
RMSE	48776.0923	48803.1359	52249.2170
MAPE	0.4511	0.4516	0.6409

Table 6.2.1 RMSE & MAPE values for Ridge, Lasso & Elastic Net Regression

Hence, it can be that out of these three statistical models, the ridge regression model has the lowest RMSE and the lowest MAPE compared to the other models.

6.3. Boosting technique Models

Under the boosting techniques, the gradient boosting technique was utilized to assess the dataset. It uses gradient descent optimization to train a sequence of weak models. Since the dataset is small, to avoid overfitting, hyperparameter tuning was used in this technique. The technique that was used to tune the model was the grid search method with cross validation and the outputs were generated as follows.

Maximum depth of each tree	Minimum number of samples in each split	Number of decision trees
4	6	50

Table 6.3.1 Hyper-parameter tuning - Gradient Boosting Regression

After tuning the parameters appropriately,, following results were made.

RMSE value	48390.8252
MAPE	0.4516

Table 6.3.2 RMSE & MAPE values - Gradient Boosting Regression

6.4. Tree Based model Techniques

It is generally accurate to say that the random forest model is efficient and less likely to overfit. Hence, using the random forest model, the following results were gained. Further, similar to the gradient boosting regression, to tune the parameters, the grid search method with cross validation was utilized.

	RMSE	MAPE
Before tuning the parameters	50335.7444	0.4399
After tuning the parameters	48481.3164	0.4625

Table 6.4.1 RMSE & MAPE values - Random Forest

7. Discussion and Conclusions

7.1. Best Model

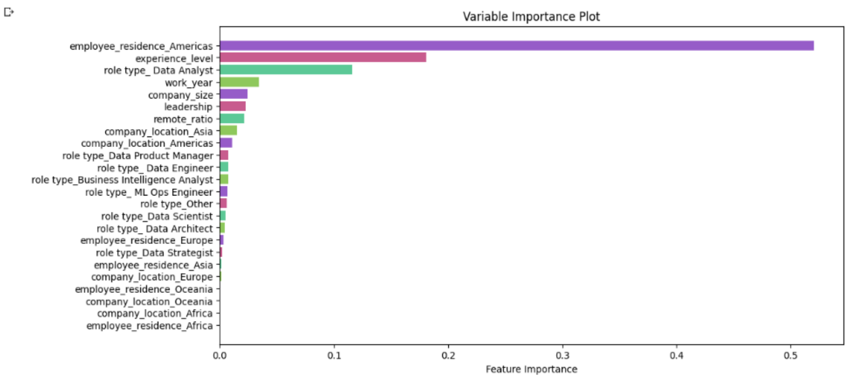
Considering all the statistical models, both the root mean square error (RMSE) and the mean absolute percentage error (MAPE) are at their lowest for the Gradient Boosting regression. Hence, the following conclusions can be drawn about the best model.

Best Model		Test	Train
Gradient Boosting Regression	RMSE	48390.8252	48390.8252
	MAPE	45126.7812	45126.7812

Table 7.1.1 Best Model

According to Table 7.1.1, it can be observed that the test RMSE is not that high when compared with the training RMSE, where a similar variation is illustrated by the MAPE values of the train and test sets. Hence, it can be concluded that the model is not overfitted.

However, when the model accuracy is considered using the R squared value, it can be noted that the proportion of variation explained by the predictors is comparatively low, resulting in $R^2 = 0.4010$.



According to Figure 7.1.1, the most significant variable has been employee_residence_Americas. The reason could be that the majority of employees that existed in the dataset are from the continent of the Americas.

Figure 7.1.1 Variable Importance Plot - Gradient Boosting Regression

In further exploration, it can be observed that experience level also has a greater impact on salaries.

The gradient boosting regression can be handled when the dataset contains a large number of categorical variables and is less prone to overfitting. Since parameters are also tuned appropriately, and the reason of supporting too many categorical variables' existence in the dataset, gradient boosting can be the best model for our dataset.

8. Issues encountered and proposed solutions.

- The variable job_title contained 83 sub-categories. Hence, using the reference https://365datascience.com/career-advice/types-of-data-science-roles-explained, further the roles were reduced to 9 sub-categories to increase the model interpretability and accuracy. However, it could be observed that classification was not 100% accurate.
- Location based variables re-categorized based on the continents to reduce newly generating amount of dummy variables.
- The best model's fit (R^2) is relatively low. Maybe a cluster analysis would suggest a better fit.

Appendix

```
import pandas as pd
df =pd.read_csv(r'D:\UOC1\3\2nd sem\ST 3082
\final project\eda\salaries_Group 5.csv')
import numpy as np
import matplotlib.pylab as plt
import seaborn as sns
df.columns
df2 =pd.read_csv(r'D:\UOC1\3\2nd sem\ST 3082\final
project\eda\all.csv')
columns = ['alpha-2','region']
df2 = df2.loc[:, columns]
df2.head()
df3 =pd.read_csv(r'D:\UOC1\3\2nd sem\ST 3082\final
project\eda\file1.csv')
df4 =pd.read_csv(r'D:\UOC1\3\2nd sem\ST 3082\final
project\eda\type.csv')
df=df.merge(df3,left_on='job_title',right_on='title',how='left')
df=df.merge(df4,left_on='type of
role',right_on='no',how='left')
df.head()
columns = ['salary_currency','salary','job_title
','Unnamed: 0','no','type of role','title']
df.drop(columns, inplace=True, axis=1)
df.head()
df=df.merge(df2,left_on='employee_residence',right_on='alph
a-2',how='left')
df.drop(['alpha-2', 'employee_residence'], inplace=True,
axis=1)
df.rename(columns={'region':'employee_residence'},
inplace=True)
df=df.merge(df2,left_on='company_location',right_on='alpha-
2',how='left')
df.drop(['alpha-2', 'company_location'], inplace=True, axis=1)
df.rename(columns={'region':'company_location'},
inplace=True)
df = df.loc[df["employment_type"] == 'FT' ]
df.drop(['employment_type'], inplace=True, axis=1)
features = ['work_year','experience_level',
'employee_residence','remote_ratio','company_location',
'company_size','leadership','role type']
x = df.loc[:, features]
y = df.loc[:, ['salary_in_usd']]
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_iris
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,
random_state=123)
df2 =pd.read_csv(r'D:\UOC1\3\2nd sem\ST 3082\final
project\eda\all.csv')
columns = ['alpha-2','region']
df2 = df2.loc[:, columns]
df3 =pd.read_csv(r'D:\UOC1\3\2nd sem\ST 3082\final
project\eda\file1.csv')
df4 =pd.read_csv(r'D:\UOC1\3\2nd sem\ST 3082\final
project\eda\type.csv')
df5 =pd.read_csv(r'D:\UOC1\3\2nd sem\ST 3082\final
project\eda\EL.csv')
df6 =pd.read_csv(r'D:\UOC1\3\2nd sem\ST 3082\final
project\eda\RR.csv')
df7 =pd.read_csv(r'D:\UOC1\3\2nd sem\ST 3082\final
project\eda\CS.csv')
df=df.merge(df3,left_on='job_title',right_on='title',how='left')
df=df.merge(df4,left_on='type of
role',right_on='no',how='left')
df = df.loc[df["employment_type"] == 'FT' ]
df.drop(['employment_type'], inplace=True, axis=1)
columns = ['salary_currency','salary','job_title',
'Unnamed: 0','no','type of role','title']
df.drop(columns, inplace=True, axis=1)
df=df.merge(df2,left_on='employee_residence',right_on='alph
a-2',how='left')
df.drop(['alpha-2', 'employee_residence'], inplace=True,
axis=1)
df.rename(columns={'region':'employee_residence'},
inplace=True)
features = ['work_year','experience_level',
'employee_residence','remote_ratio','company_location',
'company_size','leadership','role type']
x = df.loc[:, features]
y = df.loc[:, ['salary_in_usd']]
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_iris
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,
random_state=123)
print(y_train.shape)
print(x_train.shape)
#EDA
df.describe()
corr = df[['work_year','salary_in_usd']].corr()# plot the
heatmap
plt.subplots(figsize=(5,3),dpi=300)
sns.heatmap(corr, xticklabels=corr.columns,
yticklabels=corr.columns, annot=True,
cmap=sns.diverging_palette(220, 20, as_cmap=True),vmin=-
1, vmax=1)
plt.savefig('corr.png',dpi=300)
plt.show()
df.skew(axis = 0, skipna = True)
plt.figure(figsize=(5,3),dpi=300)
df['salary_in_usd'].plot(kind='hist', bins=30,
figsize=(12,6), facecolor='#c85c8e',edgecolor='black')
plt.title('Histogram of salary in usd')
plt.xlabel('Salary in USD')
plt.savefig('Freq of salary in
usd.png',dpi=300,bbox_inches='tight')
plt.show()
data=list(df.groupby(['work_year'])['work_year'].count())
plt.figure(figsize=(5,3),dpi=300)
#plt.title('Freq of year')
label1 = ['2020', '2021', '2022', '2023']
# Creating plot
plt.pie(data, labels =
label1,autopct='%1.2f%%',colors=['#965cc8','#c85c8e',
'#8ec85c','#5cc896'])
plt.title('')
# show plot
plt.savefig('Freq of year.png',dpi=300,bbox_inches='tight')
plt.show()
data=list(df.groupby(['experience_level'])
['experience_level'].count())
plt.figure(figsize=(5,3),dpi=300)
#plt.title('Freq of experience_level')
label1 = ['EN', 'EX', 'MI', 'SE']
# Creating plot
plt.pie(data, labels = label1,autopct='%1.2f%%',
colors=['#965cc8','#c85c8e','#8ec85c','#5cc896'])
plt.title('')
# show plot
plt.savefig('Freq of
experience_levely.png',dpi=300,bbox_inches='tight')
plt.show()
data=list(df.groupby(['remote_ratio'])['remote_ratio'].count(
))
plt.figure(figsize=(6,4),dpi=300)
label1 = ['No remote work', 'Partially remote work',
 'Fully remote work']
#plt.title('Freq of remote_ratio')
# Creating plot
plt.pie(data, labels =
label1,autopct='%1.2f%%',colors=['#965cc8','#c85c8e',
'#8ec85c'])
plt.title('')
# show plot
plt.savefig('Freq of
remote_ratio.png',dpi=300,bbox_inches='tight')
plt.show()
data=list(df.groupby(['leadership'])['leadership'].count())
data=list(df.groupby(['company_size'])
['company_size'].count())
plt.figure(figsize=(5,3),dpi=300)
label1 = ['L', 'M', 'S']
#plt.title('Freq of company size')
# Creating plot
plt.pie(data, labels =
label1,autopct='%1.2f%%',colors=['#965cc8',
'#c85c8e','#8ec85c'])
plt.title('')
# show plot
plt.savefig('Freq of company
size1.png',dpi=300,bbox_inches='tight')
plt.show()
plt.figure(figsize=(5,5),dpi=300)
my_colors = {' Data Architect': '#965cc8',
 'Data Scientist': '#c85c8e',
 ' Data Engineer': '#8ec85c',
'Other': '#5cc896',' Data Analyst': '#965cc8',
'Data Product Manager': '#c85c8e', 'Business
Intelligence Analyst':
 '#8ec85c','Data Strategist': '#5cc896',
' ML Ops Engineer': '#965cc8'}
sns.boxplot(data=df, x='role
type', y='salary_in_usd',palette=my_colors)
plt.xticks(rotation = 45,fontsize=5)
plt.xlabel('Role type')
plt.ylabel('Salary in USD')
plt.savefig('Salary in USD vs role type.png',dpi=300,
bbox_inches='tight')
plt.show()
plt.figure(figsize=(5,3),dpi=300)
plt.title('Salary in USD vs year')
my_colors = {2020: '#965cc8', 2021: '#c85c8e',
2022: '#8ec85c',2023: '#5cc896'}
sns.boxplot(data=df, x='work_year', y='salary_in_usd',
palette=my_colors)
plt.xlabel('Work Year')
plt.ylabel('Salary in USD')
plt.savefig('Salary in USD vs year.png',dpi=300,
bbox_inches='tight')
plt.figure(figsize=(5,3),dpi=300)
plt.title('Salary in USD vs Remote ratio')
my_colors = {0: '#965cc8', 50: '#c85c8e', 100:
'#8ec85c'}
sns.boxplot(data=df,
x='remote_ratio',
y='salary_in_usd',palette=my_colors)
plt.xlabel('Remote ratio')
plt.xticks([0,1,2], ['No remote work', 'Partially remote
work',
 'Fully remote work'],fontsize=6)
plt.ylabel('Salary in USD')
plt.savefig('Salary in USDvs Remote
ratio.png',dpi=300,
bbox_inches='tight')
plt.show()
plt.figure(figsize=(5,3),dpi=300)
plt.title('Salary in USD vs company size')
my_colors = {'M': '#965cc8', 'L': '#c85c8e', 'S':
'#8ec85c'}
sns.boxplot(data=df,
x='company_size',
y='salary_in_usd',palette=my_colors,
order=['S','M','L'])
plt.xlabel('Company size')
plt.ylabel('Salary in USD')
plt.savefig('SalaryinUSDvs
company_size.png',dpi=300,bbox_inches='tight')
plt.show()
plt.figure(figsize=(5,3),dpi=300)
plt.title('Salary in USD vs employee residence')
my_colors = {'Americas': '#965cc8', 'Europe':
'#c85c8e',
 'Oceania': '#8ec85c','Asia': '#5cc896','Africa':
'#965cc8'}
sns.boxplot(data=df,
x='employee_residence',
y='salary_in_usd',palette=my_colors)
plt.xlabel('Employee residence')
plt.ylabel('Salary in USD')
plt.savefig('SalaryinUSDvs
employee_residence.png',dpi=300,bbox_inches='tight
')
plt.show()
plt.figure(figsize=(5,3),dpi=300)
plt.title('Salary in USD vs leadership')
my_colors = {0: '#965cc8', 1: '#c85c8e'}
```

```
sns.boxplot(data=df, x='leadership', y='salary_in_usd',
palette=my_colors)
plt.xlabel('Leadership')
plt.ylabel('Salary in USD')
plt.savefig('Salary in USD vs leadership.png',dpi=300,
bbox_inches='tight')
plt.show()
plt.figure(figsize=(5,3),dpi=300)
plt.title('Salary in USD vs company location')
my_colors = {'Americas': '#965cc8', 'Europe': '#c85c8e',
 'Oceania': '#8ec85c','Asia': '#5cc896','Africa': '#965cc8'}
sns.boxplot(data=df,
x='company_location', y='salary_in_usd',palette=my_colors)
plt.xlabel('Company location')
plt.ylabel('Salary in USD')
plt.savefig('Salary in USD vs
company_location.png',dpi=300,bbox_inches='tight')
plt.show()
plt.figure(figsize=(5,3),dpi=300)
plt.title('')
my_colors = {'EN': '#965cc8', 'MI': '#c85c8e', 'SE': '#8ec85c',
'EX': '#5cc896'}
sns.boxplot(data=df,
x='work_year',
y='salary_in_usd',hue='experience_level',palette=my_colors,
hue_order=['EN','MI','SE','EX'])
plt.xlabel('Work year')
plt.ylabel('Salary in USD')
plt.savefig('Salary experience level work
year.png',dpi=300,bbox_inches='tight')
plt.show()
x=df.groupby(['work_year'])['salary_in_usd'].mean()
plt.figure(figsize=(5,3),dpi=300)
plt.plot(x,color='red',marker='*')
plt.title('Avarage salary vs Year')
plt.xticks([2020,2021,2022,2023])
plt.xlabel('Year')
plt.savefig('avgsalaryvsyear.png',dpi=300,bbox_inches='tight')
plt.show()
c_color=('#965cc8','#c85c8e','#8ec85c','#5cc896')
y=list(df.groupby(['work_year'])['salary_in_usd'].mean())
x=['2020','2021','2022','2023']
plt.figure(figsize=(5,3),dpi=300)
plt.title('Avg salary vs work year')
bars=plt.bar(x,y,color=c_color)
plt.savefig('Avg salary vs work year.png',dpi=300,figsize=(6,5))
plt.show()
df.groupby(['experience_level'])['salary_in_usd'].mean()
x=['EN','MI','SE','EX']
y=[76565.097701,103574.322896,150691.900324,200210.136364]
plt.figure(figsize=(5,3),dpi=300)
plt.title('Avg salary vs experience level')
bars=plt.bar(x,y,color=c_color)
plt.savefig('Avg salary vs experience
level.png',dpi=300,bbox_inches='tight')
plt.show()
y=list(df.groupby(['work_year'])['work_year'].count())
x=[2020,2021,2022,2023]
plt.figure(figsize=(5,3),dpi=300)
plt.title('Freq of Year')
plt.xticks([2020,2021,2022,2023])
bars=plt.bar(x,y,color=c_color)
print(x)
plt.savefig('Freq of Year1.png',dpi=300,bbox_inches='tight')
plt.show()
y=list(df.groupby(['experience_level'])['experience_level'].count())
x=['EN','EX','MI','SE']
y=[174,66,511,1545]
plt.figure(figsize=(5,3),dpi=300)
plt.title('Freq of experience level')
bars=plt.bar(x,y,color=c_color)
plt.savefig('Freq of experience level.png',dpi=300,bbox_inches='tight')
plt.show()
c_color=('#965cc8','#c85c8e','#8ec85c')
y=list(df.groupby(['remote_ratio'])['remote_ratio'].count())
x=['No remote work','Partially remote','Fully remote']
plt.figure(figsize=(5,3),dpi=300)
plt.title('Freq of remote ratio')
bars=plt.bar(x,y,color=c_color)
plt.savefig('Freq of remote ratio.png',dpi=300,bbox_inches='tight')
plt.show()
df.groupby(['company_size'])['company_size'].count()
x=['S','M','L']
```

```
mport scipy.stats as stats
#Chi-squared test statistic, sample size, and
minimum of rows and columns
X2 = stats.chi2_contingency(data_crosstab,
correction=False)[0]
minDim = min(data_crosstab.shape)-1


#calculate Cramer's V
V = np.sqrt((X2/n) / minDim)


#display Cramer's V
print(V)
data_crosstab =
pd.crosstab(x_train['employee_residence'],
                    x_train['role type'],
                       margins = False)
import scipy.stats as stats
#Chi-squared test statistic, sample size, and
minimum of rows and columns
X2 = stats.chi2_contingency(data_crosstab,
correction=False)[0]
minDim = min(data_crosstab.shape)-1


#calculate Cramer's V
V = np.sqrt((X2/n) / minDim)


#display Cramer's V
print(V)
data_crosstab =
pd.crosstab(x_train['company_location'],
                    x_train['leadership'],
                       margins = False)
import scipy.stats as stats
#Chi-squared test statistic, sample size, and
minimum of rows and columns
X2 = stats.chi2_contingency(data_crosstab,
correction=False)[0]
minDim = min(data_crosstab.shape)-1
#calculate Cramer's V
V = np.sqrt((X2/n) / minDim)


#display Cramer's V
print(V)
data_crosstab =
pd.crosstab(x_train['company_location'],
                    x_train['role type'],
                       margins = False)
import scipy.stats as stats
#Chi-squared test statistic, sample size, and
minimum of rows and columns
X2 = stats.chi2_contingency(data_crosstab,
correction=False)[0]
minDim = min(data_crosstab.shape)-1


#calculate Cramer's V
V = np.sqrt((X2/n) / minDim)


#display Cramer's V
print(V)
data_crosstab = pd.crosstab(x_train['leadership'],
                    x_train['role type'],
                       margins = False)
idata_crosstab
import scipy.stats as stats
#Chi-squared test statistic, sample size, and
minimum of rows and columns
X2 = stats.chi2_contingency(data_crosstab,
correction=False)[0]
minDim = min(data_crosstab.shape)-1


#calculate Cramer's V
V = np.sqrt((X2/n) / minDim)
#display Cramer's V
print(V)
```

```
#Onehotencoding
x_train = pd.get_dummies(x_train, columns =
['employee_residence','company_location','role
type'])
x_test = pd.get_dummies(x_test, columns =
['employee_residence','company_location','role
type'])
x_test.head()
from sklearn.linear_model import Ridge
from sklearn.linear_model import Lasso
from sklearn.linear_model import ElasticNet
from sklearn.metrics import mean_squared_error
from sklearn.metrics import
mean_absolute_percentage_error, r2_score
#Ridge
ridge = Ridge(alpha=1)
ridge.fit(x_train, y_train)
y_pred = ridge.predict(x_test)
rmse_ridge = np.sqrt(mean_squared_error(y_test,
y_pred))
mape_ridge =
mean_absolute_percentage_error(y_test, y_pred)
rmse_ridge
mape_ridge
#Lasso
mean_absolute_percentage_error, r2_score
lasso = Lasso(alpha=1)
lasso.fit(x_train, y_train)
y_pred = lasso.predict(x_test)
rmse_lasso = np.sqrt(mean_squared_error(y_test,
y_pred))
mape_lasso =
mean_absolute_percentage_error(y_test, y_pred)
rmse_lasso
mape_lasso
#Elastic Net
elastic_net = ElasticNet(alpha=1, l1_ratio=0.5)
elastic_net.fit(x_train, y_train)
y_pred = elastic_net.predict(x_test)
rmse_elastic = np.sqrt(mean_squared_error(y_test,
y_pred))
mape_elastic =
mean_absolute_percentage_error(y_test, y_pred)
rmse_elastic
Mape_elastic
y=[105,1895,296]
plt.figure(figsize=(5,3),dpi=300)
plt.title('Freq of company size')
bars=plt.bar(x,y,color=c_color)
plt.savefig('Freq of company
size.png',dpi=300,bbox_inches='tight')
plt.show()
print(pd.unique(df['employee_residence']))
y=list(df.groupby(['employee_residence'])
['employee_residence'].count())
x=['Africa','Americas','Asia','Europe','Oceania']
plt.figure(figsize=(5,3),dpi=300)
plt.title('Freq of employee residence')
bars=plt.bar(x,y,color='#c85c8e')
plt.savefig('Freq of employee
residence.png',dpi=300,bbox_inches='tight')
plt.show()
print(pd.unique(df['company_location']))
y=list(df.groupby(['company_location'])
['company_location'].count())
x=['Africa','Americas','Asia','Europe','Oceania']
plt.figure(figsize=(5,3),dpi=300)
plt.title('Freq of company location')
bars=plt.bar(x,y,color='#c85c8e')
plt.savefig('Freq of company
location.png',dpi=300,bbox_inches='tight')
plt.show()
y=list(df.groupby(['role type'])['role type'].count())
x=['Data Analyst ','Data Architect','Data Engineer','ML
Ops Engineer','Business Intelligence Analyst','Data
Product Manager','Data Scientist','Data
Strategist','Other']
plt.figure(figsize=(5,3),dpi=300)
plt.title('Freq of role type')
plt.xticks(rotation = 45)
bars=plt.bar(x,y,color='#c85c8e')
plt.savefig('Freq of role
type.png',dpi=300,bbox_inches='tight')
plt.show()
```

```
dsd=df[df["company_location"] != df["employee_residence"]]
plt.figure(figsize=(5,3),dpi=300)
sns.boxplot(data=dsd, y='salary_in_usd')
plt.ylabel('Salary in USD')
plt.savefig('diff.png',dpi=300,bbox_inches='tight')
plt.show()
data_crosstab = pd.crosstab(x_train['employee_residence'],
                            x_train['company_location'],
                              margins = False)
Data_crosstab
import scipy.stats as stats
#Chi-squared test statistic, sample size, and minimum of rows and columns
X2 = stats.chi2_contingency(data_crosstab, correction=False)[0]
n = 2296
minDim = min(data_crosstab.shape)-1


#calculate Cramer's V
V = np.sqrt((X2/n) / minDim)


#display Cramer's V
print(V)
data_crosstab = pd.crosstab(x_train['employee_residence'],
                            x_train['leadership'],
                              margins = False)
#gradient boosting method
import pandas as pd
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.metrics import accuracy_score
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_percentage_error, r2_score
model_gbm = GradientBoostingRegressor()
param_grid = {
    'n_estimators': [50, 100, 200],
    'learning_rate': [0.01, 0.1, 0.2],
    'max_depth': [3, 4, 5],
    'min_samples_split': [2, 4, 6]}
grid_search = GridSearchCV(estimator=model_gbm, param_grid=param_grid, cv=5)
best_model = grid_search.best_estimator_
best_model.fit(x_train, y_train)
y_pred = best_model.predict(x_test)
rmse_test = np.sqrt(mean_squared_error(y_test, y_pred))
mape_test = mean_absolute_percentage_error(y_test, y_pred)
r_squared_test = r2_score(y_test, y_pred)
mape_test
train_pred = best_model.predict(x_train)
rmse_train = np.sqrt(mean_squared_error(y_train, train_pred))
rmse_train
mape_train = mean_absolute_percentage_error(y_train, train_pred)
mape_train
#VIP
import matplotlib.pyplot as plt
feature_importance = best_model.feature_importances_
sorted_idx = np.argsort(feature_importance)
pos = np.arange(sorted_idx.shape[0]) + .5
plt.figure(figsize=(12, 6))
plt.barh(pos, feature_importance[sorted_idx], align='center',color=['#8ec85c','#5cc896','#c85c8e','#965cc8'])
plt.yticks(pos, np.array(x_test.columns.values.tolist())[sorted_idx])
plt.xlabel('Feature Importance')
plt.title('Variable Importance Plot')
plt.show()
```

```
#random forest
# Import the model we are using
from sklearn.ensemble import RandomForestRegressor

# Instantiate model with 1000 decision trees
rf = RandomForestRegressor(n_estimators = 1000, random_state = 42)

# Train the model on training data
rf.fit(x_train, y_train);

# Use the forest's predict method on the test data
predictions1 = rf.predict(x_test)

from sklearn.metrics import mean_squared_error
mse1 = mean_squared_error(y_test, predictions1)
print("MSE1:", mse1)
#rmse
rmse1 = np.sqrt(mse1)
print("RMSE1:", rmse1)

from sklearn.metrics import mean_absolute_percentage_error
mape1 = mean_absolute_percentage_error(y_test, predictions1)
print("MAPE1:", mape1)

from sklearn.metrics import r2_score
rSq1 = r2_score(y_test, predictions1)
print("R-squared1:", rSq1)
##hyperparameter tunning in rf
from sklearn.model_selection import RandomizedSearchCV
# Number of trees in random forest
n_estimators = [int(x) for x in np.linspace(start = 200, stop = 2000, num = 10)]
# Number of features to consider at every split
max_features = ['auto', 'sqrt']
# Maximum number of levels in tree
max_depth = [int(x) for x in np.linspace(10, 110, num = 11)]
max_depth.append(None)
# Minimum number of samples required to split a node
min_samples_split = [2, 5, 10]
# Minimum number of samples required at each leaf node
min_samples_leaf = [1, 2, 4]
# Method of selecting samples for training each tree
bootstrap = [True, False]
# Create the random grid
random_grid = {'n_estimators': n_estimators,
               'max_features': max_features,
               'max_depth': max_depth,
               'min_samples_split': min_samples_split,
               'min_samples_leaf': min_samples_leaf,
               'bootstrap': bootstrap}
print(random_grid)
#Random Search Training
#Now, we instantiate the random search and fit it like any Scikit-Learn model:
# Use the random grid to search for best hyperparameters
# First create the base model to tune
rf2 = RandomForestRegressor()

# Random search of parameters, using 3 fold cross validation,
# search across 100 different combinations, and use all available cores
rf_random = RandomizedSearchCV(estimator = rf2, param_distributions = random_grid, n_iter = 100, cv = 3, verbose=2, random_state=42, n_jobs = -1)

# Fit the random search model
rf_random.fit(x_train, y_train)
```

```
rf_random.best_params_
# Use the forest's predict method on the test data
predictions2 = rf_random.predict(x_test)

from sklearn.metrics import mean_squared_error
mse2 = mean_squared_error(y_test, predictions2)
print("MSE2:", mse2)
#rmse
rmse2 = np.sqrt(mse2)
print("RMSE2:", rmse2)

from sklearn.metrics import mean_absolute_percentage_error
mape2 = mean_absolute_percentage_error(y_test, predictions2)
print("MAPE2:", mape2)

from sklearn.metrics import r2_score
rSq2 = r2_score(y_test, predictions2)
print("R-squared2:", rSq2)
Grid Search with Cross Validation

from sklearn.model_selection import GridSearchCV
# Create the parameter grid based on the results of random search
param_grid = {
    'bootstrap': [True],
    'max_depth': [40,60,80,100],
    'max_features': [2,3],
    'min_samples_leaf': [1,2,3],
    'min_samples_split': [3,5,7],
    'n_estimators': [500,1000,1500,1800,2000]
}
# Instantiate the grid search model
grid_search = GridSearchCV(estimator = rf2, param_grid = param_grid,
                           cv = 3, n_jobs = -1, verbose = 2)

# Fit the grid search to the data
grid_search.fit(x_train, y_train)
grid_search.best_params_

best_grid = grid_search.best_estimator_
# Use the forest's predict method on the test data
predictions3 = best_grid.predict(x_test)

from sklearn.metrics import mean_squared_error
mse3 = mean_squared_error(y_test, predictions3)
print("MSE3:", mse3)
#rmse
rmse3 = np.sqrt(mse3)
print("RMSE3:", rmse3)

from sklearn.metrics import mean_absolute_percentage_error
mape3 = mean_absolute_percentage_error(y_test, predictions3)
print("MAPE3:", mape3)

from sklearn.metrics import r2_score
rSq3 = r2_score(y_test, predictions3)
print("R-squared3:", rSq3)
```