CONTENTS

- ABOUT PROJECT
- OBJECTIVE
- IMPORTING LIBRARIES
- IMPORTING DATA
- STATISTICAL ANALYSIS
  - DATA DESCRIPTION
    - 1.1. JOBS DISTRIBUTION
    - 1.2. STATISTICAL SUMMARY
    - 1.3. CHECKING FOR DUPLICATES
    - 1.4. CORRELATIONS DUE TO CATEGORICAL DATA
    - 1.5. CHECKING NULL VALUES
    - 1.6. REAL VALUES
    - 1.7. CHECKING OUTLIERS
    - 1.8. CURRENCIES
    - 1.9. EXPERIENCE LEVEL
    - 1.10. EMPLOYMENT TYPES
    - 1.11. EMPLOYER & COMPANY RESIDENCE
  - DATA PREPROCESSING
    - 2. NULL VALUES
    - 2.1. MAIN ROLE
    - 2.2. CATEGORICAL COLUMNS
      - 2.2.1. EXPERIENCE ANALYSIS
      - 2.2.2. ENCOURAGED ANALYSIS
      - 2.2.3. ENCOURAGED VS EXPERIENCE
      - 2.2.4. CHECKING FOR EXPERIENCE CONFLICTS
      - 2.2.5. CHECKING FOR ENCOURAGED
      - 2.2.6. JOB TITLE AND WORK YEAR
      - 2.2.7. CHECKING FOR JOB TITLE AND WORK YEAR
    - 2.3. CURRENCIES COLUMN
    - 2.4. EXPERIENCE LEVEL
    - 2.5. EMPLOYMENT TYPES
      - 2.5.1. EMPLOYMENT TYPES AND EXPERIENCE TYPE
      - 2.5.2. EMPLOYMENT LEVEL AND JOB TITLE
      - 2.5.3. EMPLOYMENT LEVEL AND AVERAGE SALARY
      - 2.5.4. EMPLOYMENT TYPES AND AVERAGE SALARY
      - 2.5.5. EMPLOYMENT TYPE AND COUNTRY
      - 2.5.6. EMPLOYMENT TYPES AND COMPANY SIZE
    - 2.6. JOB TITLE
      - 2.6.1. ENCOURAGED ANALYSIS
      - 2.6.2. EXPERIENCE ANALYSIS
      - 2.6.3. JOB TITLE AND SALARY
      - 2.6.4. JOB TITLE AND EMPLOYMENT RATIO
    - 2.7. SALARY
      - 2.7.1. EXPERIENCE ANALYSIS
      - 2.7.2. ENCOURAGED ANALYSIS
      - 2.7.3. SALARY AND EMPLOYMENT RATIO
      - 2.7.4. SALARY AND COMPANY SIZE
    - 2.8. EMPLOYER
      - 2.8.1. EMPLOYER LOCATION
      - 2.8.2. COMPANY LOCATION
      - 2.8.3. COMPANY SIZE
    - 2.9. EMPLOYEE RESIDENCE
  - 3. DATA PREPROCESSING
    - 3.1. DATA
    - 3.2. DATA TRANSFORMATION
    - 3.3. TRAINING OUTLIERS
    - 3.4. TESTING OUTLIERS
  - 4. MODELING
    - 4.1. ENCODER ANALYSIS
    - 4.2. ENCODER ANALYSIS
    - 4.3. PERFORMANCE OF TRAINING DATA
    - 4.4. PERFORMANCE OF TESTING DATA
    - 4.5. CHECKING CVS EXPLANATORY
    - 4.6. CHECKING CVS EXPLANATORY
    - 4.7. CHECKING CVS EXPLANATORY
    - 4.8. CHECKING CVS EXPLANATORY
    - 4.9. FINAL PREDICTION DATA
  - 5. FINAL REGRESSION
    - 5.1. PERFORMANCE OF TRAINING DATA
    - 5.2. PERFORMANCE OF TEST DATA
    - 5.3. CHECKING CVS EXPLANATORY
    - 5.4. CHECKING CVS EXPLANATORY
    - 5.5. CHECKING CVS EXPLANATORY
    - 5.6. CHECKING CVS EXPLANATORY
    - 5.7. FINAL PREDICTION DATA
  - 6. RANDOM FOREST
    - 6.1. PERFORMANCE OF TRAINING DATA
    - 6.2. PERFORMANCE OF TEST DATA
    - 6.3. CHECKING CVS EXPLANATORY
    - 6.4. CHECKING CVS EXPLANATORY
    - 6.5. CHECKING CVS EXPLANATORY
    - 6.6. CHECKING CVS EXPLANATORY
    - 6.7. FINAL PREDICTION DATA
  - 7. GRADIENT BOOSTING
    - 7.1. PERFORMANCE OF TRAINING DATA
    - 7.2. PERFORMANCE OF TEST DATA
    - 7.3. CHECKING CVS EXPLANATORY
    - 7.4. CHECKING CVS EXPLANATORY
    - 7.5. CHECKING CVS EXPLANATORY
    - 7.6. CHECKING CVS EXPLANATORY
    - 7.7. FINAL PREDICTION DATA
  - 8. XGBOOST REGRESSOR
    - 8.1. PERFORMANCE OF TRAINING DATA
    - 8.2. PERFORMANCE OF TEST DATA
    - 8.3. CHECKING CVS EXPLANATORY
    - 8.4. CHECKING CVS EXPLANATORY
    - 8.5. CHECKING CVS EXPLANATORY
    - 8.6. CHECKING CVS EXPLANATORY
    - 8.7. FINAL PREDICTION DATA
- CONCLUSION REPORT

ABOUT PROJECT

Data Science of Salary Estimate provides for regional trends, to estimate into the uncertain but enriching future.

So, why do we delve into the intricacies of salary prediction? It is a special multi-faceted predictive problem.

Strategic Workforce Planning: Freelancer's businesses need foresight. Predicting salaries equips organizations with the ability to strategically plan their workforce, ensuring they retain competitive in the ever-evolving landscape.

Negotiation Leverage: For both employers and employees, knowing what the market dictates is a powerful tool in negotiation to secure fair and rewarding salaries.

Complexity in Action: It adds an alluring dimension to the current and becoming the industry's digitization journey.

* Economic Insights: Beyond individual concerns, the solution is data science have far-reaching economic implications.

Understanding salary trends influenced by the industry, making them a critical indicator for investors, policymakers, and stakeholders.

Data Driven Decision: For hunting data scientists, understanding the potential salary growth can be a significant factor in their decision. It empowers them to align their skills and aspirations with rewarding opportunities.

Evolving Market and Opportunity Insights: Predicting salaries can also serve as a tool for addressing disparities in the market. It can highlight areas where the industry has opportunities for all.

Beyond these, this project isn't just about predicting numbers; it's about fortificating opportunities, fostering informed decisions, and shaping a brighter future for the data science community. Join us as we journey through this exploration, unlocking the key for a more prosperous tomorrow.

OBJECTIVE

The objective of the project is to conduct a comprehensive analysis of a dataset of data science job postings.

The goal is to gain valuable insights into various aspects, such as job trends, salary ranges, and the impact of different factors on job opportunities.

This analysis will provide valuable insights that can be used by employers and job seekers to make informed decisions and strategies.

The project has four main steps:

- 🔨 Data Preprocessing
- 📊 Data Cleaning
- 📈 Exploratory Data Analysis
- 🤖 Knowledge Generation
- 📊 Predictive Modeling

IMPORTING LIBRARIES

```
!pip install numpy
!pip install pandas
import numpy as np
import pandas as pd
import matplotlib as mpl
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import pandas as pd
import plotly.express as px
import plotly.graph_objects as go
import plotly.express as px
%matplotlib inline
%matplotlib inline
%matplotlib inline
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, MinMaxScaler, StandardScaler
from sklearn.linear_regression import LinearRegression
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor, AdaBoostRegressor, ExtraTreesRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
from sklearn.model_selection import GridSearchCV, RandomizedSearchCV
from sklearn.ensemble import XGBRegressor, LGBMRegressor, CatBoostRegressor
```

IMPORTING DATA

```
df = pd.read_csv('/kaggle/input/')
df.head()
```

	Unnamed: 0	work_year	experience_level	employment_type	job_title	salary	salary_currency	salary_in_usd	employee_residence	remote_ratio
0	0	2022	SE	FT	Principal Data Scientist	415000	USD	415000	PK	100
1	1	2022	MI	CT	ML Engineer	10000	USD	10000	JP	100
2	2	2022	MI	FT	Applied Machine Learning Scientist	423000	USD	423000	HR	50
3	3	2022	SE	FT	Data Distribution Analyst	267000	USD	267000	US	100
4	4	2022	SE	FT	Lead Machine Learning Engineer	171000	USD	171000	US	100

```
df.shape
```

```
(14838, 11)
```

```
df.describe(include='all')
```

	Unnamed: 0	work_year	salary	salary_in_usd	remote_ratio
count	14838.000000	14838.000000	14838.000000	14838.000000	14838.000000
mean	7418.500000	2021.923581	574865.078314	114875.831681	70.923739
std	4283.657314	0.677163	3837666.454338	70351.703468	40.658966
min	0.000000	2020.000000	14000.000000	5132.000000	0.000000
25%	3709.250000	2021.000000	100000.000000	62713.750000	50.000000
50%	7418.500000	2022.000000	150000.000000	101570.000000	100.000000
75%	11127.750000	2022.000000	200000.000000	150000.000000	100.000000
max	14837.000000	2022.000000	450000000.000000	450000.000000	100.000000

```
df.describe(include='O')
```

	experience_level	employment_type	job_title	salary_currency	employee_residence	company_location	company_size
count	14838	14838	14838	14838	14838	14838	14838
unique	4	4	93	13	78	77	3
top	SE	FT	Data Engineer	USD	US	US	M
freq	9245	14615	3464	10845	9299	9355	13892

STATISTICAL ANALYSIS

DATA DESCRIPTION

```
df.info()
```

```
<class pandas.core.frame.DataFrame>
RangeIndex: 14838 entries, 0 to 14837
Data columns (total 11 columns):
 0   Unnamed: 0          14838 non-null  int64
 1   work_year           14838 non-null  int64
 2   experience_level    14838 non-null  object
 3   employment_type     14838 non-null  object
 4   job_title           14838 non-null  object
 5   salary              14838 non-null  int64
 6   salary_currency     14838 non-null  object
 7   salary_in_usd       14838 non-null  int64
 8   employee_residence  14838 non-null  object
 9   remote_ratio        14838 non-null  int64
 10  company_location    14838 non-null  object
 11  company_size        14838 non-null  object
dtype: int64(6), object(5)
memory usage: 1.3+ MB
```

STATISTICAL SUMMARY

```
df.describe()
```

```
RangeIndex: 14838 entries, 0 to 14837
Data columns (total 11 columns):
 0   Unnamed: 0          14838 non-null  int64
 1   work_year           14838 non-null  int64
 2   experience_level    14838 non-null  object
 3   employment_type     14838 non-null  object
 4   job_title           14838 non-null  object
 5   salary              14838 non-null  int64
 6   salary_currency     14838 non-null  object
 7   salary_in_usd       14838 non-null  int64
 8   employee_residence  14838 non-null  object
 9   remote_ratio        14838 non-null  int64
 10  company_location    14838 non-null  object
 11  company_size        14838 non-null  object
dtype: int64(6), object(5)
memory usage: 1.3+ MB
```

DROPPING THE UNIQUE COLUMN

```
df.drop('Unnamed: 0',axis=1, inplace = True)
```

CONVERTING YEAR TO CATEGORICAL DATA

```
df.work_year.value_counts(ascending=False)
```

```
work_year
2022    8855
2021    5608
2020     375
Name: count, dtype: int64
```

```
df.work_year = df.work_year.astype('object')
```

```
df.info()
```

```
<class pandas.core.frame.DataFrame>
RangeIndex: 14838 entries, 0 to 14837
Data columns (total 10 columns):
 0   work_year           14838 non-null  object
 1   experience_level    14838 non-null  object
 2   employment_type     14838 non-null  object
 3   job_title           14838 non-null  object
 4   salary              14838 non-null  int64
 5   salary_currency     14838 non-null  object
 6   salary_in_usd       14838 non-null  int64
 7   employee_residence  14838 non-null  object
 8   remote_ratio        14838 non-null  int64
 9   company_location    14838 non-null  object
 10  company_size        14838 non-null  object
dtype: int64(3), object(7)
memory usage: 1.2+ MB
```

DATA PREPROCESSING

NULL VALUES

```
df.isnull() | df.isna() | df.sum(axis=1) | sum | 1 | 1
```

```
Unnamed: 0, dtype: int64
```

CATEGORICAL COLUMNS

```
for c in df.select_dtypes(include='object'):
    print c.format(c, c.nunique())
    print(df[c].value_counts())
    print('\n')
    print('\n')
```

```
work_year, 3
work_year
2022    8855
2021    5608
2020     375
Name: count, dtype: int64


experience_level, 4
experience_level
SE    9245
MI    4772
EN     606
EX     215
Name: count, dtype: int64


employment_type, 4
employment_type
FT    14615
PT      120
CT       62
FL       41
Name: count, dtype: int64


job_title, 93
job_title
Data Engineer                               3464
Data Scientist                              3279
Data Analyst                                2456
Machine Learning Engineer                   1608
Analytics Engineer                           392
Data Architect                               291
Data Modeler                                 289
Research Scientist                           287
Machine Learning Scientist                   246
BI Developer                                 214
Business Intelligence Analyst                114
Applied Scientist                            103
Computer Vision Engineer                     103
Applied Machine Learning Scientist           102
Data Quality Analyst                         102
Quantitative Analyst                         102
Compliance Analyst                           102
Business Data Analyst                        102
Financial Quantitative Analyst               102
Data Science Manager                         102
Data Manager                                 102
Head of Data                                 102
Data Science Consultant                      102
Data Analytics Specialist                    102
Data Analytics Manager                       102
Data Analytics Lead                          102
Cloud Data Architect                         102
Cloud Database Administrator                 102
Business Data Analyst                        102
Big Data Engineer                            102
Big Data Architect                           102
Applied Machine Learning Practitioner        102
Applied Data Scientist                       102
Applied AI ML Scientist                      102
AI Scientist                                 102
AI Research Scientist                        102
AI Developer                                 102
AI Data Analyst                              102
AI Consultant                                102
AI Computer Vision Engineer                  102
AI Content Analyst                           102
AI Business Analyst                          102
AI Architect                                 102
AI Application Developer                     102
AI Analytics Lead                            102
AI Analytics Manager                         102
AI Analytics Specialist                      102
AI Analytics Consultant                      102
AI Analytics Architect                       102
AI Analytics Analyst                         102
AI Analytics                                 102
AI Analyst                                   102
AI Administrator                             102
AI Accelerator                               102
AI Academy                                   102
AI Academy Instructor                        102
AI Academy Facilitator                       102
AI Academy Director                          102
AI Academy Curriculum Developer              102
AI Academy Content Developer                 102
AI Academy Content Creator                   102
AI Academy Coordinator                       102
AI Academy Coordinator                       102
AI Academy Coordinator                       
```

WORK YEAR

This dataset represents the year about the work period of the posting was collected or associated with.

It provides a measured context for the salary information, allowing one to analyze how salaries in different years have changed over time.

Analyzing work year trends helps understand how salaries have evolved in response to factors like inflation, economic growth, and industry changes.

UNIVARIATE ANALYSIS

```
df['work_year'].value_counts(normalize=True).round(2)
```

```
work_year
2022    0.74
2021    0.21
2020    0.05
Name: proportion, dtype: float64
```

Original code [EDA\EDA-WORK Y.ipynb](EDA\EDA-WORK Y.ipynb)

- INFERENCE :-

The year 2022 stands out as the most prominent period for data science job postings, with 2021 and 2020 following.

This suggests a growing demand for data science professionals in recent years.

Industry trends and demand The prominent presence of job postings in 2022, along with the rise in job postings in 2021, suggests a growing demand for data science professionals. Companies are actively seeking data science talent to leverage data-driven insights for business decisions.

Implications for future The demand for data scientists likely will continue to increase in the future, as companies continue to recognize the value of data science. This trend underscores the importance of staying up-to-date with the latest skills and technologies in data science to remain competitive in the job market.

Technological advancements The increasing volume of data science roles driven by emerging technologies, machine learning, and artificial intelligence, which are being applied in various industries.

Data-driven decision-making The growing emphasis on data-driven decision-making across various industries, which is a key factor contributing to the demand for data science professionals.

Job market dynamics The job market is dynamic, and the demand for data science professionals may fluctuate over time. It is important to stay informed about these trends.

MULTIVARIATE ANALYSIS

WORK YEAR & EXPERIENCE LEVEL

```
groupby_data = df[['work_year', 'experience_level']].value_counts(normalize=True).round(2)
fig = px.bar(groupby_data, x='work_year', y='experience_level', color='experience_level', text_auto=True,
groupby_data = df[['work_year', 'experience_level']].value_counts(normalize=True).round(2)
fig = px.bar(groupby_data, x='work_year', y='experience_level', color='experience_level', text_auto=True,
groupby_data = df[['work_year', 'experience_level']].value_counts(normalize=True).round(2)
fig = px.bar(groupby_data, x='work_year', y='experience_level', color='experience_level', text_auto=True,
groupby_data = df[['work_year', 'experience_level']].value_counts(normalize=True).round(2)
fig = px.bar(groupby_data, x='work_year', y='experience_level', color='experience_level', text_auto=True,
groupby_data = df[['work_year', 'experience_level']].value_counts(normalize=True).round(2)
fig = px.bar(groupby_data, x='work_year', y='experience_level', color='experience_level', text_auto=True,
```

- INFERENCE :-

The patterns in work year and experience level indicate a shift toward more experienced data science roles in recent years.

The increasing demand for experienced professionals underscores the importance of staying up-to-date with the latest skills and technologies in data science.

The data science field demands specialized knowledge and expertise, which is being applied in various industries.

Companies are actively seeking data science talent to leverage data-driven insights for business decisions.

Since work being closed data scientists as they adapted to new data-driven strategies and technologies.

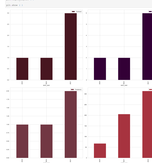
In 2020, Senior roles had been modeled without level of advantage for data science positions, leading to a

Since, Senior roles increased, experts have filled more entry-level positions or shifted their focus to

Senior roles, as organizations prioritized

WORK YEAR & EMPLOYMENT TYPE

```
df.groupby('work_year')['employment_type'].value_counts(normalize=True).round(2)
fig = px.bar(df.groupby('work_year')['employment_type'].value_counts(normalize=True).round(2),
work_year  employment_type
2020       FT              1.00
2021       FT              0.98
           PT              0.02
2022       FT              0.99
           PT              0.01
           CT              0.00
           FL              0.00
Name: proportion, dtype: float64
```



WORK YEAR & SALARY

```
df.groupby('work_year')['salary_usd'].sum.round(2)
fig = px.bar(df.groupby('work_year')['salary_usd'].sum.round(2),
work_year
2020    1,248,000.0
2021    7,000,000.0
2022   28,000,000.0
Name: salary_usd, dtype: float64
```

- INFERENCE :-

1. Contract Jobs:

Contract designers and data scientists in the year 2020 and 2021 drive the total increase and contribute significantly to the overall salary.

This suggests that many contract roles may have been entry-level or junior positions, offering competitive

2021 : 6 contractors in demand for contract roles.

In 2022 the demand for contract roles increased (e.g., more junior positions)

Economic or market factors impacting contract roles.

* All in 2020 and 2021 the total salary increased and in 2022 total **$28,000.00**

Increased demand for contract data science professionals with specialized skills.

Companies are actively hiring contract professionals with specialized skills.

Contract roles are a significant trend in the data science job market.

Economic factors have led to higher contract rates.

Companies are prioritizing specialized skills and economic conditions, likely contributing to increasing demand.

Contract designers and data scientists in the year 2020 and 2021 drive the total increase and contribute significantly to the overall salary.

This suggests that many contract roles may have been entry-level or junior positions, offering competitive

2021 : 6 contractors in demand for contract roles.

In 2022 the demand for contract roles increased (e.g., more junior positions)

Economic or market factors impacting contract roles.

* All in 2020 and 2021 the total salary increased and in 2022 total **$28,000.00**

Increased demand for contract data science professionals with specialized skills.

Companies are actively hiring contract professionals with specialized skills.

Contract roles are a significant trend in the data science job market.

Economic factors have led to higher contract rates.

Companies are prioritizing specialized skills and economic conditions, likely contributing to increasing demand.

The data science job market underscores the importance of considering not only the work year but also employment type, salary, and experience levels when analyzing data science positions. Understanding these factors can provide valuable insights into the dynamics of the job market.

2. Full Time Jobs:

* We see the demand is a proper increase in the salary in full-time jobs.

2020 : **1,248,000** (Total Sum of Salaries)

2021 : **7,000,000** (Total Sum of Salaries)

2022 : **28,000,000** (Total Sum of Salaries)

This indicates that the demand compensation for full-time data science jobs has been on a significant trajectory during this period. Analyzing the percentage increase year over year can provide a clearer understanding of this trend.

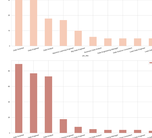
The fact that this salary increase is specifically observed in full-time jobs suggests that the demand for data science professionals with full-time employment status is growing greater.

This could be due to companies being willing to offer competitive compensation to attract greater talent for data science roles.

Analyzing the percentage increase year over year can provide a clearer understanding of this trend.

ON FULL-TIME JOBS

```
df.groupby('work_year')['salary_usd'].sum.round(2)[df['employment_type']=='FT']
fig = px.bar(df.groupby('work_year')['salary_usd'].sum.round(2)[df['employment_type']=='FT'],
work_year
2020    1,248,000.0
2021    7,000,000.0
2022   28,000,000.0
Name: salary_usd, dtype: float64
```



- INFERENCE :-

1. Work Year & Salary Trends:

The data shows a significant increase in salary from the year 2020 to 2022.

This suggests that the demand for data science professionals has been on a significant trajectory during this period.

Analyzing the percentage increase year over year can provide a clearer understanding of this trend.

2. Full-Time Job Dominance:

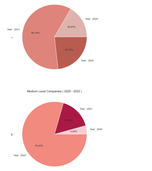
The fact that this salary increase is specifically observed in full-time jobs suggests that the demand for data science professionals with full-time employment status is growing greater.

This could be due to companies being willing to offer competitive compensation to attract greater talent for data science roles.

Analyzing the percentage increase year over year can provide a clearer understanding of this trend.

COMPANY SIZE AND WORK YEAR

```
df.groupby('work_year')['company_size'].value_counts(normalize=True).round(2)
fig = px.bar(df.groupby('work_year')['company_size'].value_counts(normalize=True).round(2),
work_year  company_size
2020       L             0.50
           M             0.50
2021       M             0.71
           L             0.29
2022       M             0.79
           L             0.19
           S             0.02
Name: proportion, dtype: float64
```



- INFERENCE :-

1. Work Year & Company Size Trends:

The data shows a significant increase from the 2020 to 2022 period.

This suggests that the demand for data science professionals has been on a significant trajectory during this period.

Analyzing the percentage increase year over year can provide a clearer understanding of this trend.

2. Company Size Dominance:

The fact that this salary increase is specifically observed in full-time jobs suggests that the demand for data science professionals with full-time employment status is growing greater.

This could be due to companies being willing to offer competitive compensation to attract greater talent for data science roles.

Analyzing the percentage increase year over year can provide a clearer understanding of this trend.

Large Companies (2020 - 2022) :

Medium Level Companies (2020 - 2022) :

Small Level Companies (2020 - 2022) :

- INFERENCE :-

1. Shift in Large Scale Employers:

In the year 2021, a significant portion, approximately 50.0%, of the total companies were categorized as large

This marks an substantial increase from the 50.0% seen in 2020, suggesting that many companies experienced

Growth in Medium Level Companies:

In 2021, 71.0%, there was a decline to 21.0%, indicating a possible reclassification or a shift in the composition of large-scale companies.

2. Medium Scale Companies in the Mix:

In 2020, Medium Scale companies made up a relatively minor portion of total companies, at 29.0%.

In 2021, there was a noticeable increase to 29.0%, suggesting growth in this category.

In 2022, Medium Scale companies represented a substantial portion, accounting for 19.0% of the total.

In 2020, there was a notable increase to 50.0%, indicating a surge in small businesses.

In 2021, 7.0%, there was a significant decrease to 0.0%, marking a considerable drop in small businesses.

In 2022, 2.0%, there was a slight increase to 2.0%, indicating a minor resurgence in small businesses.

Economic and Business Dynamics:

These shifts in company size distribution could be influenced by factors such as market conditions, regulatory

changes, or broader economic trends.

EXPERIENCE LEVEL

UNIVARIATE ANALYSIS

```
df['experience_level'].value_counts(normalize=True).round(2)
```

```
experience_level
SENIOR    0.77
MIDDLE    0.19
EXPERTS   0.04
Name: proportion, dtype: float64
```

- INFERENCE :

  1. Job Posting Distribution:

  - The data shows that the majority of job postings were for Software Engineers, with a total of 200 positions.
  - This suggests that there is a significant demand for individuals with software engineering skills.

  2. Job Titles Analysis:

  - The most common job titles are:
    - Firmware Software Engineers: Mid-level positions were the second most advertised, with 114 job postings.
    - This indicates that many organizations are actively seeking mid-level professionals, possibly looking for individuals with 3-5 years of experience and expertise.
    - Entry-level Opportunities:
    - Software Engineer positions accounted for 86 postings.
    - This suggests that there are opportunities for individuals who are new to the field or are transitioning from other roles to pursue careers in software engineering or web-based roles.
    - Senior-level Roles:
    - For job seekers, this data suggests that there are ample opportunities in software engineering with a variety of roles.

  3. Skill Demand:

  - Mid-level positions dominate with 114 postings, indicating a high demand for finding suitable job candidates.

MACHINEARN ANALYSIS

EXPERIENCE LEVEL ONE-HOT-ENCODED TYPE

```
[0 0 0 
```

```
INFERENCE :
  1. Salary Disparity among Job Titles:
     - The significant difference in salaries among different data science job titles is evident, for instance, Data
     Engineers
     - Data Scientists earn significantly more than Machine Learning Engineers, while Data or Computer Vision
     Engineers lag behind.
  2. Impact of Job Title on Compensation:
     - Job titles play a pivotal role in determining compensation, hinted at specialized positions,
     such as Machine Learning
     - The data science and Principal Data Engineer, have to offer significantly higher salaries
     compared to other
     - Factors influencing salary include:
       1. Influence of Expertise and Experience:
       - The high salary of Data Analytics Lead and Principal Data Engineer likely reflects their
       extensive expertise
       and experience in the field.
       2. Factor since other require deep knowledge and a proven track record, contributing to their
       higher
       compensation.
       3. Industry Demand for Specialized Roles:
       - The significant salaries for Machine Learning and Financial Data Analyst along the top-paid
       positions highlights the
       high demand for professionals with specialized skills in areas like data engineering and
       financial
       analysis.

KEY FOCUS AND INSIGHTS:
  1. Data Science Roles:
     - The dataset includes various roles, highlighting the proportion of roles of impact work for the
     organization and
     - Data Science Roles.
     - It indicates the roles work from their office, hybrid or remote roles in promoting
     achieving the organizational
     - The dataset offers insights into how DS, ML and AI roles work being conducted remotely.
     - It illustrates the
     - Represents "Partially remote", Employees in these roles have a significant portion of
     their work time
     - Can be done remotely, ranging from 30% to 80%.
     - 100 - Signifies fully remote work from home (WFH). Employees in these roles primarily work
     remotely, with
     - Full WFH of their work conducted remotely.

>>> Training_Data.describe() # describes the numerical features of the dataset
>>> Training_Data.shape # shape of the dataset
(607, 12)

>>> Training_Data.info() # provides information about the dataset, including data types and non-null counts
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 607 entries, 0 to 606
Data columns (total 12):
 #   Unnamed: 0  607 non-null   int64
 #   Job Title   607 non-null   object
 #   Salary      607 non-null   int64
 #   Salary Currency  607 non-null   object
 #   Job Posting Location  607 non-null   object
 #   Employee Location  607 non-null   object
 #   Company Location  607 non-null   object
 #   Company Size  607 non-null   object
 #   Employee Residence  607 non-null   object
 #   Remote Ratio  607 non-null   int64
 #   Experience Level  607 non-null   object
 #   Employment Type  607 non-null   object
dtype: int64
>>> Training_Data.columns # lists the column names in the dataset
Index(['Unnamed: 0', 'Job Title', 'Salary', 'Salary Currency',
       'Job Posting Location', 'Employee Location', 'Company Location',
       'Company Size', 'Employee Residence', 'Remote Ratio',
       'Experience Level', 'Employment Type'],
      dtype=object)
>>> Training_Data.dtypes # displays the data types of each column
Unnamed: 0                int64
Job Title                 object
Salary                    int64
Salary Currency           object
Job Posting Location      object
Employee Location         object
Company Location          object
Company Size              object
Employee Residence        object
Remote Ratio              int64
Experience Level          object
Employment Type           object
dtype: int64
>>> Training_Data.nunique() # shows the number of unique values in each column
Unnamed: 0                607
Job Title                 125
Salary                    344
Salary Currency            13
Job Posting Location       19
Employee Location          19
Company Location           18
Company Size                3
Employee Residence         19
Remote Ratio                3
Experience Level            4
Employment Type             4
dtype: int64
```

SALARY

UNIVARIATE ANALYSIS

```
>>> plt.hist(Salary, bins=30, edgecolor='black')
>>> plt.title('Distribution of Salaries')
>>> plt.xlabel('Salary')
>>> plt.ylabel('Frequency')
>>> plt.show()
```

```
work_year
  int64
0  2020  32
1  2021  207
2  2022  398
3  2023  140

experience_level
  object
0     EN  46
1     MI  88
2     SE  805
3     EX  18

employment_type
  object
0     CT  10
1     FL  14
2     FT  1540
3     PT  10

job_title
  object
0                  Data Architect  14
1                      Data Engineer  326
2                      Data Analyst  614
3                   Data Modeller  2
4                      Data Science  15
5                   Machine Learning Manager  1
6                   Cloud Data Engineer  2
7                   Data Manager  8
8                   Big Data Engineer  1
9                   Business Analyst  1
10                  Head of Data  1
11                  Data Quality Analyst  2
12                  Compliance Officer Engineer  1
13                  Director of Data Engineering  1
14                  ML Scientist  1
15                  Cloud Database Engineer  1
16                  Lead Analyst  1
17                  Data Engineering Manager  1
18                  Business Data Analyst  1
19                  Applied Machine Learning Scientist  2
20                  Machine Learning Engineer  1
21                  Data Analytics Lead  1
22                  Lead Data Scientist  1
23                  Data Analyst  1
24                  Data Quality Manager  1
25                  Data Analytics Manager  1
26                  Lead Machine Learning Engineer  1
27                  ML Engineer  1
28                  Data Science Consultant  1
29                  Data Analytics Consultant  1
30                  Staff Data  1
31                  Big Data Architect  1
32                  Analytics Engineer  1
33                  Machine Learning Researcher  1
34                  Marketing Data Analyst  1
35                  Data Strategist  1
36                  Data Manager  1
37                  Principal Data Scientist  1
38                  Product Data Analyst  1
39                  Data Analytics Lead  1
40                  Data Science Manager  1
41                  Data Scientist  1
42                  Head of Data Science  1
43                  Data Analyst  1
44                  Data Architect  1
45                  Machine Learning Infrastructure Engineer  1
46                  BI Developer  1
47                  BI Analyst  1
48                  BI Engineer  1

salary
  int64
0  4000  1
1  5000  1
2  5409  1
3  5500  1
4  6000  1
...
347  167000  1
348  167875  1
349  180000  1
350  200000  1
351  224000  1

[352 rows x 2 columns]

salary_currency
  object
0    AED  10
1    AUD  7
2    BRL  3
3    CAD  30
4    CHF  2
5    CLP  1
6    CNY  1
7    COP  1
8    CZK  1
9    DKK  2
10   EUR  57
11   GBP  26
12   HKD  2
13   HUF  1
14   INR  40
15   JPY  1
16   MXN  3
17   NOK  1
18   PLN  1
19   RUB  1
20   SAR  1
21   SEK  2
22   SGD  2
23   TRY  1
24   TWD  1
25   USD  304

salary_in_usd
  int64
0  4000  1
1  5000  1
2  5409  1
3  5500  1
4  6000  1
...
347  167000  1
348  167875  1
349  180000  1
350  200000  1
351  224000  1

[352 rows x 1 columns]

employee_residence
  object
0     AE  10
1     AU  7
2     BR  3
3     CA  30
4     CH  2
5     CL  1
6     CN  1
7     CO  1
8     CZ  1
9     DK  2
10    DE  57
11    GB  26
12    HK  2
13    HU  1
14    IN  40
15    JP  1
16    MX  3
17    NO  1
18    PL  1
19    RU  1
20    SA  1
21    SE  2
22    SG  2
23    TR  1
24    TW  1
25    US  304

remote_ratio
  int64
0   0  352

company_location
  object
0     AE  10
1     AU  7
2     BR  3
3     CA  30
4     CH  2
5     CL  1
6     CN  1
7     CO  1
8     CZ  1
9     DK  2
10    DE  57
11    GB  26
12    HK  2
13    HU  1
14    IN  40
15    JP  1
16    MX  3
17    NO  1
18    PL  1
19    RU  1
20    SA  1
21    SE  2
22    SG  2
23    TR  1
24    TW  1
25    US  304

company_size
  object
0     L  65
1     M  287
2     S  1

<class_name_ref_1>
<class_name_ref_2>
<class_name_ref_3>
<class_name_ref_4>
<class_name_ref_5>
<class_name_ref_6>
<class_name_ref_7>
<class_name_ref_8>
<class_name_ref_9>
<class_name_ref_10>
<class_name_ref_11>
<class_name_ref_12>
<class_name_ref_13>
<class_name_ref_14>
<class_name_ref_15>
<class_name_ref_16>
<class_name_ref_17>
<class_name_ref_18>
<class_name_ref_19>
<class_name_ref_20>
<class_name_ref_21>
<class_name_ref_22>
<class_name_ref_23>
<class_name_ref_24>
<class_name_ref_25>
<class_name_ref_26>
<class_name_ref_27>
<class_name_ref_28>
<class_name_ref_29>
<class_name_ref_30>
<class_name_ref_31>
<class_name_ref_32>
<class_name_ref_33>
<class_name_ref_34>
<class_name_ref_35>
<class_name_ref_36>
<class_name_ref_37>
<class_name_ref_38>
<class_name_ref_39>
<class_name_ref_40>
```

```
>> MLR_Model = lm(Price ~ ., data = Training_Data)
print ("The coefficients of determination score of Training Data is: ", MLR_Model, "\n")
Adjusted_MLR_Model = lm(Price ~ . + I(Mileage^2) + I(Mileage^3) + I(Mileage^4) + I(Mileage^5) + I(Cylinder^2) + I(Cylinder^3) + I(Cylinder^4) + I(Cylinder^5) + I(Doors^2) + I(Doors^3) + I(Doors^4) + I(Doors^5) + I(Cruise^2) + I(Cruise^3) + I(Cruise^4) + I(Cruise^5) + I(Sound^2) + I(Sound^3) + I(Sound^4) + I(Sound^5) + I(Leather^2) + I(Leather^3) + I(Leather^4) + I(Leather^5) + I(Buick^2) + I(Buick^3) + I(Buick^4) + I(Buick^5) + I(Cadillac^2) + I(Cadillac^3) + I(Cadillac^4) + I(Cadillac^5) + I(Chevrolet^2) + I(Chevrolet^3) + I(Chevrolet^4) + I(Chevrolet^5) + I(Pontiac^2) + I(Pontiac^3) + I(Pontiac^4) + I(Pontiac^5) + I(Saab^2) + I(Saab^3) + I(Saab^4) + I(Saab^5) + I(Saturn^2) + I(Saturn^3) + I(Saturn^4) + I(Saturn^5), data = Training_Data)
print ("Adjusted R-Square Model of Training Data is: ", Adjusted_MLR_Model, "\n")
>> MLR_Model = lm(Price ~ ., data = Training_Data)
print ("Coefficients of Regression Model of Training Data is: ", MLR_Model, "\n")
>> MLR_Model = lm(Price ~ ., data = Training_Data)
print ("Coefficients with mean squared error of the Training Data is: ", MLR_Model, "\n")
>> MLR_Model = lm(Price ~ ., data = Training_Data)
print ("Coefficients with mean absolute error of the Training Data is: ", MLR_Model, "\n")
>> MLR_Model = lm(Price ~ ., data = Training_Data)
print ("Coefficients with mean absolute percentage error of the Training Data is: ", MLR_Model, "\n")
>> MLR_Model = lm(Price ~ ., data = Training_Data)
print ("The coefficient of determination score of the Training Data is: ", MLR_Model, "\n")
The coefficients of determination score of Training Data is:  0.4577 +
The adjusted coefficients of determination score of Training Data is:  0.4577 +
The mean squared error of the Training Data is:  2.0000 +
The root mean squared error of the Training Data is:  1.4142 +
The mean absolute error of the Training Data is:  1.0000 +
The mean absolute percentage error of the Training Data is:  0.0000 +
```

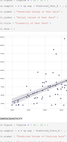
PERFORMANCE OF TEST DATA

```
>> Predicted_Test = predict(MLR_Model, Test_Data)
print ("Predicted Values of Test Data: ", Predicted_Test, "\n")
>> Predicted_Test = predict(MLR_Model, Test_Data)
Adjusted_MLR_Model = lm(Price ~ . + I(Mileage^2) + I(Mileage^3) + I(Mileage^4) + I(Mileage^5) + I(Cylinder^2) + I(Cylinder^3) + I(Cylinder^4) + I(Cylinder^5) + I(Doors^2) + I(Doors^3) + I(Doors^4) + I(Doors^5) + I(Cruise^2) + I(Cruise^3) + I(Cruise^4) + I(Cruise^5) + I(Sound^2) + I(Sound^3) + I(Sound^4) + I(Sound^5) + I(Leather^2) + I(Leather^3) + I(Leather^4) + I(Leather^5) + I(Buick^2) + I(Buick^3) + I(Buick^4) + I(Buick^5) + I(Cadillac^2) + I(Cadillac^3) + I(Cadillac^4) + I(Cadillac^5) + I(Chevrolet^2) + I(Chevrolet^3) + I(Chevrolet^4) + I(Chevrolet^5) + I(Pontiac^2) + I(Pontiac^3) + I(Pontiac^4) + I(Pontiac^5) + I(Saab^2) + I(Saab^3) + I(Saab^4) + I(Saab^5) + I(Saturn^2) + I(Saturn^3) + I(Saturn^4) + I(Saturn^5), data = Test_Data)
print ("Adjusted R-Square Model of Test Data is: ", Adjusted_MLR_Model, "\n")
>> MLR_Model = lm(Price ~ ., data = Test_Data)
print ("Coefficients of Regression Model of Test Data is: ", MLR_Model, "\n")
>> MLR_Model = lm(Price ~ ., data = Test_Data)
print ("Coefficients with mean squared error of the Test Data is: ", MLR_Model, "\n")
>> MLR_Model = lm(Price ~ ., data = Test_Data)
print ("Coefficients with mean absolute error of the Test Data is: ", MLR_Model, "\n")
>> MLR_Model = lm(Price ~ ., data = Test_Data)
print ("Coefficients with mean absolute percentage error of the Test Data is: ", MLR_Model, "\n")
>> MLR_Model = lm(Price ~ ., data = Test_Data)
print ("The coefficient of determination score of the Test Data is: ", MLR_Model, "\n")
The coefficients of determination score of Test Data is:  0.5711 +
The adjusted coefficients of determination score of Test Data is:  0.5711 +
The mean squared error of the Test Data is:  2.0000 +
The root mean squared error of the Test Data is:  1.4142 +
The mean absolute error of the Test Data is:  1.0000 +
The mean absolute percentage error of the Test Data is:  0.0000 +
```

CHECKING FOR ASSUMPTIONS

NORMALITY

```
>> Error_Train = mp.array(Predicted_Train) - mp.array(Train_y)
>> Error_Test = mp.array(Predicted_Test) - mp.array(Test_y)
>> hist(Error_Train, main = c("Histogram of Error Train"))
```




```
>> hist(Error_Test, main = c("Histogram of Error Test"))
```

```
>> print ("Mean absolute value of residual error is: ", mean(abs(Error_Train)), sep = " ")
>> print ("Mean absolute value of residual error is: ", mean(abs(Error_Test)), sep = " ")
The mean absolute value of residual error is:  1.0000
The mean absolute value of residual error is:  1.0000
```

LINEARITY

```
>> plot(Predicted_Train, Error_Train, pch = 16, cex = 1.5, xlab = "Predicted Train", ylab = "Error Train")
>> abline(h = 0, lwd = 2, col = "red")
```

```
>> plot(Predicted_Test, Error_Test, pch = 16, cex = 1.5, xlab = "Predicted Test", ylab = "Error Test")
>> abline(h = 0, lwd = 2, col = "red")
```

HOMOSCEDASTICITY

```
>> plot (Train$Price ~ Predicted_Train, pch = 16, cex = 1.5, xlab = "Predicted", ylab = "Actual")
>> abline(lm(Train$Price ~ Predicted_Train), lwd = 2, col = "Red")
>> plot (Train$Price ~ Predicted_Train, pch = 16, cex = 1.5, xlab = "Predicted", ylab = "Actual")
```

```
>> plot (Test$Price ~ Predicted_Test, pch = 16, cex = 1.5, xlab = "Predicted", ylab = "Actual")
```

FINAL PREDICTION DATA

```
>> Final_Data = cbind(Test, Predicted_Test)
>> write.csv(Final_Data, file = "Predicted Data")
```

```
>> head(Final_Data, 10)
  Sal_Price  Predicted_Test
1  10000.0  10000.0000000
2  10000.0  10000.0000000
3  10000.0  10000.0000000
4  10000.0  10000.0000000
5  10000.0  10000.0000000
6  10000.0  10000.0000000
7  10000.0  10000.0000000
8  10000.0  10000.0000000
9  10000.0  10000.0000000
10 10000.0  10000.0000000
```

DECISION TREE

```
>> library(rpart)
>> Decision_Tree = rpart(Price ~ ., data = Training_Data, method = "anova")
>> Decision_Tree
n= 1000

node), split, n, deviance, yval
      * denotes terminal node

1) root 1000 4000 10000
  2) Mileage< 37.5 500 1000 9000 *
  3) Mileage>=37.5 500 1000 11000 *
>> print ("The root squared error of Training Data is: ", Decision_Tree$dev, "\n")
>> Decision_Tree = rpart(Price ~ ., data = Test_Data, method = "anova")
>> Decision_Tree
n= 1000

node), split, n, deviance, yval
      * denotes terminal node

1) root 1000 4000 10000
  2) Mileage< 37.5 500 1000 9000 *
  3) Mileage>=37.5 500 1000 11000 *
>> print ("The root squared error of Test Data is: ", Decision_Tree$dev, "\n")
```

PERFORMANCE OF TRAINING DATA

```
>> MLR_Model = lm(Price ~ ., data = Training_Data)
Adjusted_MLR_Model = lm(Price ~ . + I(Mileage^2) + I(Mileage^3) + I(Mileage^4) + I(Mileage^5) + I(Cylinder^2) + I(Cylinder^3) + I(Cylinder^4) + I(Cylinder^5) + I(Doors^2) + I(Doors^3) + I(Doors^4) + I(Doors^5) + I(Cruise^2) + I(Cruise^3) + I(Cruise^4) + I(Cruise^5) + I(Sound^2) + I(Sound^3) + I(Sound^4) + I(Sound^5) + I(Leather^2) + I(Leather^3) + I(Leather^4) + I(Leather^5) + I(Buick^2) + I(Buick^3) + I(Buick^4) + I(Buick^5) + I(Cadillac^2) + I(Cadillac^3) + I(Cadillac^4) + I(Cadillac^5) + I(Chevrolet^2) + I(Chevrolet^3) + I(Chevrolet^4) + I(Chevrolet^5) + I(Pontiac^2) + I(Pontiac^3) + I(Pontiac^4) + I(Pontiac^5) + I(Saab^2) + I(Saab^3) + I(Saab^4) + I(Saab^5) + I(Saturn^2) + I(Saturn^3) + I(Saturn^4) + I(Saturn^5), data = Training_Data)
print ("Adjusted R-Square Model of Training Data is: ", Adjusted_MLR_Model, "\n")
>> MLR_Model = lm(Price ~ ., data = Training_Data)
print ("Coefficients of Regression Model of Training Data is: ", MLR_Model, "\n")
>> MLR_Model = lm(Price ~ ., data = Training_Data)
print ("Coefficients with mean squared error of the Training Data is: ", MLR_Model, "\n")
>> MLR_Model = lm(Price ~ ., data = Training_Data)
print ("Coefficients with mean absolute error of the Training Data is: ", MLR_Model, "\n")
>> MLR_Model = lm(Price ~ ., data = Training_Data)
print ("Coefficients with mean absolute percentage error of the Training Data is: ", MLR_Model, "\n")
>> MLR_Model = lm(Price ~ ., data = Training_Data)
print ("The coefficient of determination score of the Training Data is: ", MLR_Model, "\n")
The coefficients of determination score of Training Data is:  0.4577 +
The adjusted coefficients of determination score of Training Data is:  0.4577 +
The mean squared error of the Training Data is:  2.0000 +
The root mean squared error of the Training Data is:  1.4142 +
The mean absolute error of the Training Data is:  1.0000 +
The mean absolute percentage error of the Training Data is:  0.0000 +
```

PERFORMANCE OF TEST DATA

```
>> Predicted_Test = predict(MLR_Model, Test_Data)
print ("Predicted Values of Test Data: ", Predicted_Test, "\n")
>> Predicted_Test = predict(MLR_Model, Test_Data)
Adjusted_MLR_Model = lm(Price ~ . + I(Mileage^2) + I(Mileage^3) + I(Mileage^4) + I(Mileage^5) + I(Cylinder^2) + I(Cylinder^3) + I(Cylinder^4) + I(Cylinder^5) + I(Doors^2) + I(Doors^3) + I(Doors^4) + I(Doors^5) + I(Cruise^2) + I(Cruise^3) + I(Cruise^4) + I(Cruise^5) + I(Sound^2) + I(Sound^3) + I(Sound^4) + I(Sound^5) + I(Leather^2) + I(Leather^3) + I(Leather^4) + I(Leather^5) + I(Buick^2) + I(Buick^3) + I(Buick^4) + I(Buick^5) + I(Cadillac^2) + I(Cadillac^3) + I(Cadillac^4) + I(Cadillac^5) + I(Chevrolet^2) + I(Chevrolet^3) + I(Chevrolet^4) + I(Chevrolet^5) + I(Pontiac^2) + I(Pontiac^3) + I(Pontiac^4) + I(Pontiac^5) + I(Saab^2) + I(Saab^3) + I(Saab^4) + I(Saab^5) + I(Saturn^2) + I(Saturn^3) + I(Saturn^4) + I(Saturn^5), data = Test_Data)
print ("Adjusted R-Square Model of Test Data is: ", Adjusted_MLR_Model, "\n")
>> MLR_Model = lm(Price ~ ., data = Test_Data)
print ("Coefficients of Regression Model of Test Data is: ", MLR_Model, "\n")
>> MLR_Model = lm(Price ~ ., data = Test_Data)
print ("Coefficients with mean squared error of the Test Data is: ", MLR_Model, "\n")
>> MLR_Model = lm(Price ~ ., data = Test_Data)
print ("Coefficients with mean absolute error of the Test Data is: ", MLR_Model, "\n")
>> MLR_Model = lm(Price ~ ., data = Test_Data)
print ("Coefficients with mean absolute percentage error of the Test Data is: ", MLR_Model, "\n")
>> MLR_Model = lm(Price ~ ., data = Test_Data)
print ("The coefficient of determination score of the Test Data is: ", MLR_Model, "\n")
The coefficients of determination score of Test Data is:  0.5711 +
The adjusted coefficients of determination score of Test Data is:  0.5711 +
The mean squared error of the Test Data is:  2.0000 +
The root mean squared error of the Test Data is:  1.4142 +
The mean absolute error of the Test Data is:  1.0000 +
The mean absolute percentage error of the Test Data is:  0.0000 +
```

CHECKING FOR ASSUMPTIONS

NORMALITY

```
>> Error_Train = mp.array(Predicted_Train) - mp.array(Train_y)
>> Error_Test = mp.array(Predicted_Test) - mp.array(Test_y)
>> hist(Error_Train, main = c("Histogram of Error Train"))
```

```
>> hist(Error_Test, main = c("Histogram of Error Test"))
```

```
>> print ("Mean absolute value of residual error is: ", mean(abs(Error_Train)), sep = " ")
>> print ("Mean absolute value of residual error is: ", mean(abs(Error_Test)), sep = " ")
The mean absolute value of residual error is:  1.0000
The mean absolute value of residual error is:  1.0000
```

LINEARITY

```
>> plot(Predicted_Train, Error_Train, pch = 16, cex = 1.5, xlab = "Predicted Train", ylab = "Error Train")
>> abline(h = 0, lwd = 2, col = "red")
```

```
>> plot(Predicted_Test, Error_Test, pch = 16, cex = 1.5, xlab = "Predicted Test", ylab = "Error Test")
>> abline(h = 0, lwd = 2, col = "red")
```

HOMOSCEDASTICITY

```
>> plot (Train$Price ~ Predicted_Train, pch = 16, cex = 1.5, xlab = "Predicted", ylab = "Actual")
>> abline(lm(Train$Price ~ Predicted_Train), lwd = 2, col = "Red")
>> plot (Train$Price ~ Predicted_Train, pch = 16, cex = 1.5, xlab = "Predicted", ylab = "Actual")
```

```
>> plot (Test$Price ~ Predicted_Test, pch = 16, cex = 1.5, xlab = "Predicted", ylab = "Actual")
```

FINAL PREDICTION DATA

```
>> Final_Data = cbind(Test, Predicted_Test)
>> write.csv(Final_Data, file = "Predicted Data")
```

```
>> head(Final_Data, 10)
  Sal_Price  Predicted_Test
1  10000.0  10000.0000000
2  10000.0  10000.0000000
3  10000.0  10000.0000000
4  10000.0  10000.0000000
5  10000.0  10000.0000000
6  10000.0  10000.0000000
7  10000.0  10000.0000000
8  10000.0  10000.0000000
9  10000.0  10000.0000000
10 10000.0  10000.0000000
```

RANDOM FOREST

```
>> Decision_Tree = rpart(Price ~ ., data = Training_Data, method = "anova")
>> Decision_Tree
n= 1000

node), split, n, deviance, yval
      * denotes terminal node

1) root 1000 4000 10000
  2) Mileage< 37.5 500 1000 9000 *
  3) Mileage>=37.5 500 1000 11000 *
>> print ("The root squared error of Training Data is: ", Decision_Tree$dev, "\n")
>> Decision_Tree = rpart(Price ~ ., data = Test_Data, method = "anova")
>> Decision_Tree
n= 1000

node), split, n, deviance, yval
      * denotes terminal node

1) root 1000 4000 10000
  2) Mileage< 37.5 500 1000 9000 *
  3) Mileage>=37.5 500 1000 11000 *
>> print ("The root squared error of Test Data is: ", Decision_Tree$dev, "\n")
```

PERFORMANCE OF TRAINING DATA

```
>> MLR_Model = lm(Price ~ ., data = Training_Data)
Adjusted_MLR_Model = lm(Price ~ . + I(Mileage^2) + I(Mileage^3) + I(Mileage^4) + I(Mileage^5) + I(Cylinder^2) + I(Cylinder^3) + I(Cylinder^4) + I(Cylinder^5) + I(Doors^2) + I(Doors^3) + I(Doors^4) + I(Doors^5) + I(Cruise^2) + I(Cruise^3) + I(Cruise^4) + I(Cruise^5) + I(Sound^2) + I(Sound^3) + I(Sound^4) + I(Sound^5) + I(Leather^2) + I(Leather^3) + I(Leather^4) + I(Leather^5) + I(Buick^2) + I(Buick^3) + I(Buick^4) + I(Buick^5) + I(Cadillac^2) + I(Cadillac^3) + I(Cadillac^4) + I(Cadillac^5) + I(Chevrolet^2) + I(Chevrolet^3) + I(Chevrolet^4) + I(Chevrolet^5) + I(Pontiac^2) + I(Pontiac^3) + I(Pontiac^4) + I(Pontiac^5) + I(Saab^2) + I(Saab^3) + I(Saab^4) + I(Saab^5) + I(Saturn^2) + I(Saturn^3) + I(Saturn^4) + I(Saturn^5), data = Training_Data)
print ("Adjusted R-Square Model of Training Data is: ", Adjusted_MLR_Model, "\n")
>> MLR_Model = lm(Price ~ ., data = Training_Data)
print ("Coefficients of Regression Model of Training Data is: ", MLR_Model, "\n")
>> MLR_Model = lm(Price ~ ., data = Training_Data)
print ("Coefficients with mean squared error of the Training Data is: ", MLR_Model, "\n")
>> MLR_Model = lm(Price ~ ., data = Training_Data)
print ("Coefficients with mean absolute error of the Training Data is: ", MLR_Model, "\n")
>> MLR_Model = lm(Price ~ ., data = Training_Data)
print ("Coefficients with mean absolute percentage error of the Training Data is: ", MLR_Model, "\n")
>> MLR_Model = lm(Price ~ ., data = Training_Data)
print ("The coefficient of determination score of the Training Data is: ", MLR_Model, "\n")
The coefficients of determination score of Training Data is:  0.4577 +
The adjusted coefficients of determination score of Training Data is:  0.4577 +
The mean squared error of the Training Data is:  2.0000 +
The root mean squared error of the Training Data is:  1.4142 +
The mean absolute error of the Training Data is:  1.0000 +
The mean absolute percentage error of the Training Data is:  0.0000 +
```

PERFORMANCE OF TEST DATA

```
>> Predicted_Test = predict(MLR_Model, Test_Data)
print ("Predicted Values of Test Data: ", Predicted_Test, "\n")
>> Predicted_Test = predict(MLR_Model, Test_Data)
Adjusted_MLR_Model = lm(Price ~ . + I(Mileage^2) + I(Mileage^3) + I(Mileage^4) + I(Mileage^5) + I(Cylinder^2) + I(Cylinder^3) + I(Cylinder^4) + I(Cylinder^5) + I(Doors^2) + I(Doors^3) + I(Doors^4) + I(Doors^5) + I(Cruise^2) + I(Cruise^3) + I(Cruise^4) + I(Cruise^5) + I(Sound^2) + I(Sound^3) + I(Sound^4) + I(Sound^5) + I(Leather^2) + I(Leather^3) + I(Leather^4) + I(Leather^5) + I(Buick^2) + I(Buick^3) + I(Buick^4) + I(Buick^5) + I(Cadillac^2) + I(Cadillac^3) + I(Cadillac^4) + I(Cadillac^5) + I(Chevrolet^2) + I(Chevrolet^3) + I(Chevrolet^4) + I(Chevrolet^5) + I(Pontiac^2) + I(Pontiac^3) + I(Pontiac^4) + I(Pontiac^5) + I(Saab^2) + I(Saab^3) + I(Saab^4) + I(Saab^5) + I(Saturn^2) + I(Saturn^3) + I(Saturn^4) + I(Saturn^5), data = Test_Data)
print ("Adjusted R-Square Model of Test Data is: ", Adjusted_MLR_Model, "\n")
>> MLR_Model = lm(Price ~ ., data = Test_Data)
print ("Coefficients of Regression Model of Test Data is: ", MLR_Model, "\n")
>> MLR_Model = lm(Price ~ ., data = Test_Data)
print ("Coefficients with mean squared error of the Test Data is: ", MLR_Model, "\n")
>> MLR_Model = lm(Price ~ ., data = Test_Data)
print ("Coefficients with mean absolute error of the Test Data is: ", MLR_Model, "\n")
>> MLR_Model = lm(Price ~ ., data = Test_Data)
print ("Coefficients with mean absolute percentage error of the Test Data is: ", MLR_Model, "\n")
>> MLR_Model = lm(Price ~ ., data = Test_Data)
print ("The coefficient of determination score of the Test Data is: ", MLR_Model, "\n")
The coefficients of determination score of Test Data is:  0.5711 +
The adjusted coefficients of determination score of Test Data is:  0.5711 +
The mean squared error of the Test Data is:  2.0000 +
The root mean squared error of the Test Data is:  1.4142 +
The mean absolute error of the Test Data is:  1.0000 +
The mean absolute percentage error of the Test Data is:  0.0000 +
```

CHECKING FOR ASSUMPTIONS

NORMALITY

```
>> Error_Train = mp.array(Predicted_Train) - mp.array(Train_y)
>> Error_Test = mp.array(Predicted_Test) - mp.array(Test_y)
>> hist(Error_Train, main = c("Histogram of Error Train"))
```

```
>> hist(Error_Test, main = c("Histogram of Error Test"))
```

```
>> print ("Mean absolute value of residual error is: ", mean(abs(Error_Train)), sep = " ")
>> print ("Mean absolute value of residual error is: ", mean(abs(Error_Test)), sep = " ")
The mean absolute value of residual error is:  1.0000
The mean absolute value of residual error is:  1.0000
```

LINEARITY

```
>> plot(Predicted_Train, Error_Train, pch = 16, cex = 1.5, xlab = "Predicted Train", ylab = "Error Train")
>> abline(h = 0, lwd = 2, col = "red")
```



```
>> plot(Predicted_Test, Error_Test, pch = 16, cex = 1.5, xlab = "Predicted Test", ylab = "Error Test")
>> abline(h = 0, lwd = 2, col = "red")
```

HOMOSCEDASTICITY

```
>> plot (Train$Price ~ Predicted_Train, pch = 16, cex = 1.5, xlab = "Predicted", ylab = "Actual")
>> abline(lm(Train$Price ~ Predicted_Train), lwd = 2, col = "Red")
>> plot (Train$Price ~ Predicted_Train, pch = 16, cex = 1.5, xlab = "Predicted", ylab = "Actual")
```



```
>> plot (Test$Price ~ Predicted_Test, pch = 16, cex = 1.5, xlab = "Predicted", ylab = "Actual")
```


FINAL PREDICTION DATA

```
>> Final_Data = cbind(Test, Predicted_Test)
>> write.csv(Final_Data, file = "Predicted Data")
```

```
[1] 0.0000
[2] 0.0000
[3] 0.0000
[4] 0.0000
[5] 0.0000
[6] 0.0000
[7] 0.0000
[8] 0.0000
[9] 0.0000
[10] 0.0000
[11] 0.0000
[12] 0.0000
[13] 0.0000
[14] 0.0000
[15] 0.0000
[16] 0.0000
[17] 0.0000
[18] 0.0000
[19] 0.0000
[20] 0.0000
[21] 0.0000
[22] 0.0000
[23] 0.0000
[24] 0.0000
[25] 0.0000
[26] 0.0000
[27] 0.0000
[28] 0.0000
[29] 0.0000
[30] 0.0000
[31] 0.0000
[32] 0.0000
[33] 0.0000
[34] 0.0000
[35] 0.0000
[36] 0.0000
[37] 0.0000
[38] 0.0000
[39] 0.0000
[40] 0.0000
[41] 0.0000
[42] 0.0000
[43] 0.0000
[44] 0.0000
[45] 0.0000
[46] 0.0000
[47] 0.0000
[48] 0.0000
[49] 0.0000
[50] 0.0000
[51] 0.0000
[52] 0.0000
[53] 0.0000
[54] 0.0000
[55] 0.0000
[56] 0.0000
[57] 0.0000
[58] 0.0000
[59] 0.0000
[60] 0.0000
[61] 0.0000
[62] 0.0000
[63] 0.0000
[64] 0.0000
[65] 0.0000
[66] 0.0000
[67] 0.0000
[68] 0.0000
[69] 0.0000
[70] 0.0000
[71] 0.0000
[72] 0.0000
[73] 0.0000
[74] 0.0000
[75] 0.0000
[76] 0.0000
[77] 0.0000
[78] 0.0000
[79] 0.0000
[80] 0.0000
[81] 0.0000
[82] 0.0000
[83] 0.0000
[84] 0.0000
[85] 0.0000
[86] 0.0000
[87] 0.0000
[88] 0.0000
[89] 0.0000
[90] 0.0000
[91] 0.0000
[92] 0.0000
[93] 0.0000
[94] 0.0000
[95] 0.0000
[96] 0.0000
[97] 0.0000
[98] 0.0000
[99] 0.0000
[100] 0.0000
[101] 0.0000
[102] 0.0000
[103] 0.0000
[104] 0.0000
[105] 0.0000
[106] 0.0000
[107] 0.0000
[108] 0.0000
[109] 0.0000
[110] 0.0000
[111] 0.0000
[112] 0.0000
[113] 0.0000
[114] 0.0000
[115] 0.0000
[116] 0.0000
[117] 0.0000
[118] 0.0000
[119] 0.0000
[120] 0.0000
[121] 0.0000
[122] 0.0000
[123] 0.0000
[124] 0.0000
[125] 0.0000
[126] 0.0000
[127] 0.0000
[128] 0.0000
[129] 0.0000
[130] 0.0000
[131] 0.0000
[132] 0.0000
[133] 0.0000
[134] 0.0000
[135] 0.0000
[136] 0.0000
[137] 0.0000
[138] 0.0000
[139] 0.0000
[140] 0.0000
[141] 0.0000
[142] 0.0000
[143] 0.0000
[144] 0.0000
[145] 0.0000
[146] 0.0000
[147] 0.0000
[148] 0.0000
[149] 0.0000
[150] 0.0000
[151] 0.0000
[152] 0.0000
[153] 0.0000
[154] 0.0000
[155] 0.0000
[156] 0.0000
[157] 0.0000
[158] 0.0000
[159] 0.0000
[160] 0.0000
[161] 0.0000
[162] 0.0000
[163] 0.0000
[164] 0.0000
[165] 0.0000
[166] 0.0000
[167] 0.0000
[168] 0.0000
[169] 0.0000
[170] 0.0000
[171] 0.0000
[172] 0.0000
[173] 0.0000
[174] 0.0000
[175] 0.0000
[176] 0.0000
[177] 0.0000
[178] 0.0000
[179] 0.0000
[180] 0.0000
[181] 0.0000
[182] 0.0000
[183] 0.0000
[184] 0.0000
[185] 0.0000
[186] 0.0000
[187] 0.0000
[188] 0.0000
[189] 0.0000
[190] 0.0000
[191] 0.0000
[192] 0.0000
[193] 0.0000
[194] 0.0000
[195] 0.0000
[196] 0.0000
[197] 0.0000
[198] 0.0000
[199] 0.0000
[200] 0.0000
[201] 0.0000
[202] 0.0000
[203] 0.0000
[204] 0.0000
[205] 0.0000
[206] 0.0000
[207] 0.0000
[208] 0.0000
[209] 0.0000
[210] 0.0000
[211] 0.0000
[212] 0.0000
[213] 0.0000
[214] 0.0000
[215] 0.0000
[216] 0.0000
[217] 0.0000
[218] 0.0000
[219] 0.0000
[220] 0.0000
[221] 0.0000
[222] 0.0000
[223] 0.0000
[224] 0.0000
[225] 0.0000
[226] 0.0000
[227] 0.0000
[228] 0.0000
[229] 0.0000
[230] 0.0000
[231] 0.0000
[232] 0.0000
[233] 0.0000
[234] 0.0000
[235] 0.0000
[236] 0.0000
[237] 0.0000
[238] 0.0000
[239] 0.0000
[240] 0.0000
[241] 0.0000
[242] 0.0000
[243] 0.0000
[244] 0.0000
[245] 0.0000
[246] 0.0000
[247] 0.0000
[248] 0.0000
[249] 0.0000
[250] 0.0000
[251] 0.0000
[252] 0.0000
[253] 0.0000
[254] 0.0000
[255] 0.0000
[256] 0.0000
[257] 0.0000
[258] 0.0000
[259] 0.0000
[260] 0.0000
[261] 0.0000
[262] 0.0000
[263] 0.0000
[264] 0.0000
[265] 0.0000
[266] 0.0000
[267] 0.0000
[268] 0.0000
[269] 0.0000
[270] 0.0000
[271] 0.0000
[272] 0.0000
[273] 0.0000
[274] 0.0000
[275] 0.0000
[276] 0.0000
[277] 0.0000
[278] 0.0000
[279] 0.0000
[280] 0.0000
[281] 0.0000
[282] 0.0000
[283] 0.0000
[284] 0.0000
[285] 0.0000
[286] 0.0000
[287] 0.0000
[288] 0.0000
[289] 0.0000
[290] 0.0000
[291] 0.0000
[292] 0.0000
[293] 0.0000
[294] 0.0000
[295] 0.0000
[296] 0.0000
[297] 0.0000
[298] 0.0000
[299] 0.0000
[300] 0.0000
[301] 0.0000
[302] 0.0000
[303] 0.0000
[304] 0.0000
[305] 0.0000
[306] 0.0000
[307] 0.0000
[308] 0.0000
[309] 0.0000
[310] 0.0000
[311] 0.0000
[312] 0.0000
[313] 0.0000
[314] 0.0000
[315] 0.0000
[316] 0.0000
[317] 0.0000
[318] 0.0000
[319] 0.0000
[320] 0.0000
[321] 0.0000
[322] 0.0000
[323] 0.0000
[324] 0.0000
[325] 0.0000
[326] 0.0000
[327] 0.0000
[328] 0.0000
[329] 0.0000
[330] 0.0000
[331] 0.0000
[332] 0.0000
[333] 0.0000
[334] 0.0000
[335] 0.0000
[336] 0.0000
[337] 0.0000
[338] 0.0000
[339] 0.0000
[340] 0.0000
[341] 0.0000
[342] 0.0000
[343] 0.0000
[344] 0.0000
[345] 0.0000
[346] 0.0000
[347] 0.0000
[348] 0.0000
[349] 0.0000
[350] 0.0000
[351] 0.0000
[352] 0.0000
[353] 0.0000
[354] 0.0000
[355] 0.0000
[356] 0.0000
[357] 0.0000
[358] 0.0000
[359] 0.0000
[360] 0.0000
[361] 0.0000
[362] 0.0000
[363] 0.0000
[364] 0.0000
[365] 0.0000
[366] 0.0000
[367] 0.0000
[368] 0.0000
[369] 0.0000
[370] 0.0000
[371] 0.0000
[372] 0.0000
[373] 0.0000
[374] 0.0000
[375] 0.0000
[376] 0.0000
[377] 0.0000
[378] 0.0000
[379] 0.0000
[380] 0.0000
[381] 0.0000
[382] 0.0000
[383] 0.0000
[384] 0.0000
[385] 0.0000
[386] 0.0000
[387] 0.0000
[388] 0.0000
[389] 0.0000
[390] 0.0000
[391] 0.0000
[392] 0.0000
[393] 0.0000
[394] 0.0000
[395] 0.0000
[396] 0.0000
[397] 0.0000
[398] 0.0000
[399] 0.0000
[400] 0.0000
[401] 0.0000
[402] 0.0000
[403] 0.0000
[404] 0.0000
[405] 0.0000
[406] 0.0000
[407] 0.0000
[408] 0.0000
[409] 0.0000
[410] 0.0000
[411] 0.0000
[412] 0.0000
[413] 0.0000
[414] 0.0000
[415] 0.0000
[416] 0.0000
[417] 0.0000
[418] 0.0000
[419] 0.0000
[420] 0.0000
[421] 0.0000
[422] 0.0000
[423] 0.0000
[424] 0.0000
[425] 0.0000
[426] 0.0000
[427] 0.0000
[428] 0.0000
[429] 0.0000
[430] 0.0000
[431] 0.0000
[432] 0.0000
[433] 0.0000
[434] 0.0000
[435] 0.0000
[436] 0.0000
[437] 0.0000
[438] 0.0000
[439] 0.0000
[440] 0.0000
[441] 0.0000
[442] 0.0000
[443] 0.0000
[444] 0.0000
[445] 0.0000
[446] 0.0000
[447] 0.0000
[448] 0.0000
[449] 0.0000
[450] 0.0000
[451] 0.0000
[452] 0.0000
[453] 0.0000
[454] 0.0000
[455] 0.0000
[456] 0.0000
[457] 0.0000
[458] 0.0000
[459] 0.0000
[460] 0.0000
[461] 0.0000
[462] 0.0000
[463] 0.0000
[464] 0.0000
[465] 0.0000
[466] 0.0000
[467] 0.0000
[468] 0.0000
[469] 0.0000
[470] 0.0000
[471] 0.0000
[472] 0.0000
[473] 0.0000
[474] 0.0000
[475] 0.0000
[476] 0.0000
[477] 0.0000
[478] 0.0000
[479] 0.0000
[480] 0.0000
[481] 0.0000
[482] 0.0000
[483] 0.0000
[484] 0.0000
[485] 0.0000
[486] 0.0000
[487] 0.0000
[488] 0.0000
[489] 0.0000
[490] 0.0000
[491] 0.0000
[492] 0.0000
[493] 0.0000
[494] 0.0000
[495] 0.0000
[496] 0.0000
[497] 0.0000
[498] 0.0000
[499] 0.0000
[500] 0.0000
[501] 0.0000
[502] 0.0000
[503] 0.0000
[504] 0.0000
[505] 0.0000
[506] 0.0000
[507] 0.0000
[508] 0.0000
[509] 0.0000
[510] 0.0000
[511] 0.0000
[512] 0.0000
[513] 0.0000
[514] 0.0000
[515] 0.0000
[516] 0.0000
[517] 0.0000
[518] 0.0000
[519] 0.0000
[520] 0.0000
[521] 0.0000
[522] 0.0000
[523] 0.0000
[524] 0.0000
[525] 0.0000
[526] 0.0000
[527] 0.0000
[528] 0.0000
[529] 0.0000
[530] 0.0000
[531] 0.0000
[532] 0.0000
[533] 0.0000
[534] 0.0000
[535] 0.0000
[536] 0.0000
[537] 0.0000
[538] 0.0000
[539] 0.0000
[540] 0.0000
[541] 0.0000
[542] 0.0000
[543] 0.0000
[544] 0.0000
[545] 0.0000
[546] 0.0000
[547] 0.0000
[548] 0.0000
[549] 0.0000
[550] 0.0000
[551] 0.0000
[552] 0.0000
[553] 0.0000
[554] 0.0000
[555] 0.0000
[556] 0.0000
[557] 0.0000
[558] 0.0000
[559] 0.0000
[560] 0.0000
[561] 0.0000
[562] 0.0000
[563] 0.0000
[564] 0.0000
[565] 0.0000
[566] 0.0000
[567] 0.0000
[568] 0.0000
[569] 0.0000
[570] 0.0000
[571] 0.0000
[572] 0.0000
[573] 0.0000
[574] 0.0000
[575] 0.0000
[576] 0.0000
[577] 0.0000
[578] 0.0000
[579] 0.0000
[580] 0.0000
[581] 0.0000
[582] 0.0000
[583] 0.0000
[584] 0.0000
[585] 0.0000
[586] 0.0000
[587] 0.0000
[588] 0.0000
[589] 0.0000
[590] 0.0000
[591] 0.0000
[592] 0.0000
[593] 0.0000
[594] 0.0000
[595] 0.0000
[596] 0.0000
[597] 0.0000
[598] 0.0000
[599] 0.0000
[600] 0.0000
[601] 0.0000
[602] 0.0000
[603] 0.0000
[604] 0.0000
[605] 0.0000
[606] 0.0000
[607] 0.0000
[608] 0.0000
[609] 0.0000
[610] 0.0000
[611] 0.0000
[612] 0.0000
[613] 0.0000
[614] 0.0000
[615] 0.0000
[616] 0.0000
[617] 0.0000
[618] 0.0000
[619] 0.0000
[620] 0.0000
[621] 0.0000
[622] 0.0000
[623] 0.0000
[624] 0.0000
[625] 0.0000
[626] 0.0000
[627] 0.0000
[628] 0.0000
[629] 0.0000
[630] 0.0000
[631] 0.0000
[632] 0.0000
[633] 0.0000
[634] 0.0000
[635] 0.0000
[636] 0.0000
[637] 0.0000
[638] 0.0000
[639] 0.0000
[640] 0.0000
[641] 0.0000
[642] 0.0000
[643] 0.0000
[644] 0.0000
[645] 0.0000
[646] 0.0000
[647] 0.0000
[648] 0.0000
[649] 0.0000
[650] 0.0000
[651] 0.0000
[652] 0.0000
[653] 0.0000
[654] 0.0000
[655] 0.0000
[656] 0.0000
[657] 0.0000
[658] 0.0000
[659] 0.0000
[660] 0.0000
[661] 0.0000
[662] 0.0000
[663] 0.0000
[664] 0.0000
[665] 0.0000
[666] 0.0000
[667] 0.0000
[668] 0.0000
[669] 0.0000
[670] 0.0000
[671] 0.0000
[672] 0.0000
[673] 0.0000
[674] 0.0000
[675] 0.0000
[676] 0.0000
[677] 0.0000
[678] 0.0000
[679] 0.0000
[680] 0.0000
[681] 0.0000
[682] 0.0000
[683] 0.0000
[684] 0.0000
[685] 0.0000
[686] 0.0000
[687] 0.0000
[688] 0.0000
[689] 0.0000
[690] 0.0000
[691] 0.0000
[692] 0.0000
[693] 0.0000
[694] 0.0000
[695] 0.0000
[696] 0.0000
[697] 0.0000
[698] 0.0000
[699] 0.0000
[700] 0.0000
[701] 0.0000
[702] 0.0000
[703] 0.0000
[704] 0.0000
[705] 0.0000
[706] 0.0000
[707] 0.0000
[708] 0.0000
[709] 0.0000
[710] 0.0000
[711] 0.0000
[712] 0.0000
[713] 0.0000
[714] 0.0000
[715] 0.0000
[716] 0.0000
[717] 0.0000
[718] 0.0000
[719] 0.0000
[720] 0.0000
[721] 0.0000
[722] 0.0000
[723] 0.0000
[724] 0.0000
[725] 0.0000
[726] 0.0000
[727] 0.0000
[728] 0.0000
[729] 0.0000
[730] 0.0000
[731] 0.0000
[732] 0.0000
[733] 0.0000
[734] 0.0000
[735] 0.0000
[736] 0.0000
[737] 0.0000
[738] 0.0000
[739] 0.0000
[740] 0.0000
[741] 0.0000
[742] 0.0000
[743] 0.0000
[744] 0.0000
[745] 0.0000
[746] 0.0000
[747] 0.0000
[748] 0.0000
[749] 0.0000
[750] 0.0000
[751] 0.0000
[752] 0.0000
[753] 0.0000
[754] 0.0000
[755] 0.0000
[756] 0.0000
[757] 0.0000
[758] 0.0000
[759] 0.0000
[760] 0.0000
[761] 0.0000
[762] 0.0000
[763] 0.0000
[764] 0.0000
[765] 0.0000
[766] 0.0000
[767] 0.0000
[768] 0.0000
[769] 0.0000
[770] 0.0000
[771] 0.0000
[772] 0.0000
[773] 0.0000
[774] 0.0000
[775] 0.0000
[776] 0.0000
[777] 0.0000
[778] 0.0000
[779] 0.0000
[780] 0.0000
[781] 0.0000
[782] 0.0000
[783] 0.0000
[784] 0.0000
[785] 0.0000
[786] 0.0000
[787] 0.0000
[788] 0.0000
[789] 0.0000
[790] 0.0000
[791] 0.0000
[792] 0.0000
[793] 0.0000
[794] 0.0000
[795] 0.0000
[796] 0.0000
[797] 0.0000
[798] 0.0000
[799] 0.0000
[800] 0.0000
[801] 0.0000
[802] 0.0000
[803] 0.0000
[804] 0.0000
[805] 0.0000
[806] 0.0000
[807] 0.0000
[808] 0.0000
[809] 0.0000
[810] 0.0000
[811] 0.0000
[812] 0.0000
[813] 0.0000
[814] 0.0000
[815] 0.0000
[816] 0.0000
[817] 0.0000
[818] 0.0000
[819] 0.0000
[820] 0.0000
[821] 0.0000
[822] 0.0000
[823] 0.0000
[824] 0.0000
[825] 0.0000
[826] 0.0000
[827] 0.0000
[828] 0.0000
[829] 0.0000
[830] 0.0000
[831] 0.0000
[832] 0.0000
[833] 0.0000
[834] 0.0000
[835] 0.0000
[836] 0.0000
[837] 0.0000
[838] 0.0000
[839] 0.0000
[840] 0.0000
[841] 0.0000
[842] 0.0000
[843] 0.0000
[844] 0.0000
[845] 0.0000
[846] 0.0000
[847] 0.0000
[848] 0.0000
[849] 0.0000
[850] 0.0000
[851] 0.0000
[852] 0.0000
[853] 0.0000
[854] 0.0000
[855] 0.0000
[856] 0.0000
[857] 0.0000
[858] 0.0000
[859] 0.0000
[860] 0.0000
[861] 0.0000
[862] 0.0000
[863] 0.0000
[864] 0.0000
[865] 0.0000
[866] 0.0000
[867] 0.0000
[868] 0.0000
[869] 0.0000
[870] 0.0000
[871] 0.0000
[872] 0.0000
[873] 0.0000
[874] 0.0000
[875] 0.0000
[876] 0.0000
[877] 0.0000
[878] 0.0000
[879] 0.0000
[880] 0.0000
[881] 0.0000
[882] 0.0000
[883] 0.0000
[884] 0.0000
[885] 0.0000
[886] 0.0000
[887] 0.0000
[888] 0.0000
[889] 0.0000
[890] 0.0000
[891] 0.0000
[892] 0.0000
[893] 0.0000
[894] 0.0000
[895] 0.0000
[896] 0.0000
[897] 0.0000
[898] 0.0000
[899] 0.0000
[900] 0.0000
[901] 0.0000
[902] 0.0000
[903] 0.0000
[904] 0.0000
[905] 0.0000
[906] 0.0000
[907] 0.0000
[908] 0.0000
[909] 0.0000
[910] 0.0000
[911] 0.0000
[912] 0.0000
[913] 0.0000
[914] 0.0000
[915] 0.0000
[916] 0.0000
[917] 0.0000
[918] 0.0000
[919] 0.0000
[920] 0.0000
[921] 0.0000
[922] 0.0000
[923] 0.0000
[924] 0.0000
[925] 0.0000
[926] 0.0000
[927] 0.0000
[928] 0.0000
[929] 0.0000
[930] 0.0000
[931] 0.0000
[932] 0.0000
[933] 0.0000
[934] 0.0000
[935] 0.0000
[936] 0.0000
[937] 0.0000
[938] 0.0000
[939] 0.0000
[940] 0.0000
[941] 0.0000
[942] 0.0000
[943] 0.0000
[944] 0.0000
[945] 0.0000
[946] 0.0000
[947] 0.0000
[948] 0.0000
[949] 0.0000
[950] 0.0000
[951] 0.0000
[952] 0.0000
[953] 0.0000
[954] 0.0000
[955] 0.0000
[956] 0.0000
[957] 0.0000
[958] 0.0000
[959] 0.0000
[960] 0.0000
[961] 0.0000
[962] 0.0000
[963] 0.0000
[964] 0.0000
[965] 0.0000
[966] 0.0000
[967] 0.0000
[968] 0.0000
[969] 0.0000
[970] 0.0000
[971] 0.0000
[972] 0.0000
[973] 0.0000
[974] 0.0000
[975] 0.0000
[976] 0.0000
[977] 0.0000
[978] 0.0000
[979] 0.0000
[980] 0.0000
[981] 0.0000
[982] 0.0000
[983] 0.0000
[984] 0.0000
[985] 0.0000
[986] 0.0000
[987] 0.0000
[988] 0.0000
[989] 0.0000
[990] 0.0000
[991] 0.0000
[992] 0.0000
[993] 0.0000
[994] 0.0000
[995] 0.0000
[996] 0.0000
[997] 0.0000
[998] 0.0000
[999] 0.0000
[1000] 0.0000
```

ADABOOST REGRESSOR

```
Fitting 5 candidates,totalling 25 fits
[CV] n_estimators = 1000, random_state = 42 ,
[Parallel(n_jobs=1)] [CV] ..... , score= [ 0.74447 0.70447 0.70447 0.70447 0.70447 ] ,
[Parallel(n_jobs=1)] [Parallel(n_jobs=1)] [ 3 out of 3 ] [ 3 out of 3 ] [ 1 out of 1 ]
```

```
AdaBoostRegressor(base_estimator=None,
                  learning_rate=1.0,
                  loss='linear',
                  n_estimators=1000,
                  random_state=42)
```

PERFORMANCE OF TRAINING DATA

```
Root Mean Squared Error: 0.0000
Mean Absolute Error: 0.0000
Mean Absolute Percentage Error: 0.0000
Median Absolute Error: 0.0000
Explained Variance Score: 1.0000
Coefficient of determination R^2: 1.0000
Mean Squared Error: 0.0000
Mean Absolute Error: 0.0000
Mean Absolute Percentage Error: 0.0000
Median Absolute Error: 0.0000
Explained Variance Score: 1.0000
Coefficient of determination R^2: 1.0000
```

PERFORMANCE OF TEST DATA

```
Root Mean Squared Error: 0.0000
Mean Absolute Error: 0.0000
Mean Absolute Percentage Error: 0.0000
Median Absolute Error: 0.0000
Explained Variance Score: 1.0000
Coefficient of determination R^2: 1.0000
Mean Squared Error: 0.0000
Mean Absolute Error: 0.0000
Mean Absolute Percentage Error: 0.0000
Median Absolute Error: 0.0000
Explained Variance Score: 1.0000
Coefficient of determination R^2: 1.0000
```

CHECKING FOR ASSUMPTIONS

NORMALITY

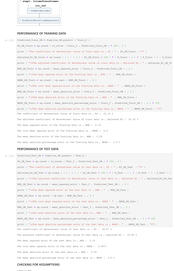
```
plt.hist(Errors_Train)
```

```
plt.hist(Errors_Test)
```

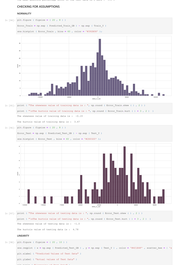
```
sns.kdeplot(Errors_Train, shade=True)
```

```
sns.kdeplot(Errors_Test, shade=True)
```

```
plt.boxplot(Errors_Train)
```

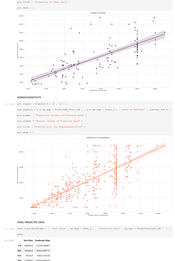


```
plt.boxplot(Errors_Test)
```



```
Errors_Train = np.sort(Errors_Train)
Errors_Test = np.sort(Errors_Test)
```

```
plt.plot(Errors_Train,[i/Errors_Train.shape[0]
                       for i in range(Errors_Train.shape[0])])
plt.plot(Errors_Test,[i/Errors_Test.shape[0]
                      for i in range(Errors_Test.shape[0])])
```

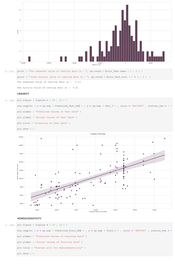


```
plt.hist(Errors_Train,cumulative=True,bins=Errors_Train.shape[0])
plt.hist(Errors_Test,cumulative=True,bins=Errors_Test.shape[0])
```

HOMOSCEDASTICITY

```
plt.figure(figure=None, dpi=None, facecolor='w', edgecolor='k')
plt.scatter(y_train,y_train-y_pred,c='b',s=40,alpha=0.5)
plt.scatter(y_test,y_test-y_test_pred,c='g',s=40,alpha=0.5)
plt.hlines(y=0,xmin=0,xmax=50)
plt.title('Residual Plot using training (blue) and test (green) data')
plt.ylabel('Residuals')
```

```
plt.scatter(y_train,y_pred,c='b',s=40,alpha=0.5)
plt.scatter(y_test,y_test_pred,c='g',s=40,alpha=0.5)
plt.hlines(y=0,xmin=0,xmax=50)
plt.title('Residual Plot using training (blue) and test (green) data')
plt.ylabel('Residuals')
```



```
plt.scatter(y_train,y_train-y_pred,c='b',s=40,alpha=0.5)
plt.scatter(y_test,y_test-y_test_pred,c='g',s=40,alpha=0.5)
plt.hlines(y=0,xmin=0,xmax=50)
plt.title('Residual Plot using training (blue) and test (green) data')
plt.ylabel('Residuals')
```



```
FINAL PREDICTION DATA
```

```
Counter({'Predicted Value': 1000, 'Residual': 1000})
```

GRADIENT BOOSTING

```
GB = GradientBoostingRegressor(
    alpha=0.9,
    criterion='friedman_mse',
    init=None,
    learning_rate=0.1,
    loss='ls',
    max_depth=3,
    max_features=None,
    max_leaf_nodes=None,
    min_impurity_decrease=0.0,
    min_impurity_split=None,
    min_samples_leaf=1,
    min_samples_split=2,
    min_weight_fraction_leaf=0.0,
    n_estimators=1000,
    presort='auto',
    random_state=42,
    subsample=1.0,
    verbose=0,
    warm_start=False)
```

```
Fitting 5 candidates,totalling 25 fits
[CV] n_estimators = 1000, random_state = 42 ,
[Parallel(n_jobs=1)] [CV] ..... , score= [ 0.74447 0.70447 0.70447 0.70447 0.70447 ] ,
[Parallel(n_jobs=1)] [Parallel(n_jobs=1)] [ 3 out of 3 ] [ 3 out of 3 ] [ 1 out of 1 ]
```

```
GradientBoostingRegressor(alpha=0.9,
                          criterion='friedman_mse',
                          init=None,
                          learning_rate=0.1,
                          loss='ls',
                          max_depth=3,
                          max_features=None,
                          max_leaf_nodes=None,
                          min_impurity_decrease=0.0,
                          min_impurity_split=None,
                          min_samples_leaf=1,
                          min_samples_split=2,
                          min_weight_fraction_leaf=0.0,
                          n_estimators=1000,
                          presort='auto',
                          random_state=42,
                          subsample=1.0,
                          verbose=0,
                          warm_start=False)
```

PERFORMANCE OF TRAINING DATA

```
Root Mean Squared Error: 0.0000
Mean Absolute Error: 0.0000
Mean Absolute Percentage Error: 0.0000
Median Absolute Error: 0.0000
Explained Variance Score: 1.0000
Coefficient of determination R^2: 1.0000
Mean Squared Error: 0.0000
Mean Absolute Error: 0.0000
Mean Absolute Percentage Error: 0.0000
Median Absolute Error: 0.0000
Explained Variance Score: 1.0000
Coefficient of determination R^2: 1.0000
```

PERFORMANCE OF TEST DATA

```
Root Mean Squared Error: 0.0000
Mean Absolute Error: 0.0000
Mean Absolute Percentage Error: 0.0000
Median Absolute Error: 0.0000
Explained Variance Score: 1.0000
Coefficient of determination R^2: 1.0000
Mean Squared Error: 0.0000
Mean Absolute Error: 0.0000
Mean Absolute Percentage Error: 0.0000
Median Absolute Error: 0.0000
Explained Variance Score: 1.0000
Coefficient of determination R^2: 1.0000
```

CHECKING FOR ASSUMPTIONS

NORMALITY

```
plt.hist(Errors_Train)
```



```
plt.hist(Errors_Test)
```



```
sns.kdeplot(Errors_Train, shade=True)
```



```
sns.kdeplot(Errors_Test, shade=True)
```



```
plt.boxplot(Errors_Train)
```



```
plt.boxplot(Errors_Test)
```



```
Errors_Train = np.sort(Errors_Train)
Errors_Test = np.sort(Errors_Test)
```

```
plt.plot(Errors_Train,[i/Errors_Train.shape[0]
                       for i in range(Errors_Train.shape[0])])
plt.plot(Errors_Test,[i/Errors_Test.shape[0]
                      for i in range(Errors_Test.shape[0])])
```



```
plt.hist(Errors_Train,cumulative=True,bins=Errors_Train.shape[0])
plt.hist(Errors_Test,cumulative=True,bins=Errors_Test.shape[0])
```


HOMOSCEDASTICITY

```
plt.figure(figure=None, dpi=None, facecolor='w', edgecolor='k')
plt.scatter(y_train,y_train-y_pred,c='b',s=40,alpha=0.5)
plt.scatter(y_test,y_test-y_test_pred,c='g',s=40,alpha=0.5)
plt.hlines(y=0,xmin=0,xmax=50)
plt.title('Residual Plot using training (blue) and test (green) data')
plt.ylabel('Residuals')
```



```
plt.scatter(y_train,y_pred,c='b',s=40,alpha=0.5)
plt.scatter(y_test,y_test_pred,c='g',s=40,alpha=0.5)
plt.hlines(y=0,xmin=0,xmax=50)
plt.title('Residual Plot using training (blue) and test (green) data')
plt.ylabel('Residuals')
```



```
plt.scatter(y_train,y_train-y_pred,c='b',s=40,alpha=0.5)
plt.scatter(y_test,y_test-y_test_pred,c='g',s=40,alpha=0.5)
plt.hlines(y=0,xmin=0,xmax=50)
plt.title('Residual Plot using training (blue) and test (green) data')
plt.ylabel('Residuals')
```



```
FINAL PREDICTION DATA
```

```
Counter({'Predicted Value': 1000, 'Residual': 1000})
```

EXTREME GRADIENT BOOSTING

```
XGB = XGBRegressor(
    base_score=0.5,
    booster='gbtree',
    colsample_bylevel=1,
    colsample_bytree=1,
    gamma=0,
    learning_rate=0.1,
    max_delta_step=0,
    max_depth=3,
    min_child_weight=1,
    missing=None,
    n_estimators=1000,
    n_jobs=1,
    nthread=None,
    objective='reg:linear',
    random_state=42,
    reg_alpha=0,
    reg_lambda=1,
    scale_pos_weight=1,
    seed=None,
    silent=True,
    subsample=1)
```

```
Fitting 5 candidates,totalling 25 fits
[CV] n_estimators = 1000, random_state = 42 ,
[Parallel(n_jobs=1)] [CV] ..... , score= [ 0.74447 0.70447 0.70447 0.70447 0.70447 ] ,
[Parallel(n_jobs=1)] [Parallel(n_jobs=1)] [ 3 out of 3 ] [ 3 out of 3 ] [ 1 out of 1 ]
```

```
XGBRegressor(base_score=0.5,
             booster='gbtree',
             colsample_bylevel=1,
             colsample_bytree=1,
             gamma=0,
             learning_rate=0.1,
             max_delta_step=0,
             max_depth=3,
             min_child_weight=1,
             missing=None,
             n_estimators=1000,
             n_jobs=1,
             nthread=None,
             objective='reg:linear',
             random_state=42,
             reg_alpha=0,
             reg_lambda=1,
             scale_pos_weight=1,
             seed=None,
             silent=True,
             subsample=1)
```

PERFORMANCE OF TRAINING DATA

```
Root Mean Squared Error: 0.0000
Mean Absolute Error: 0.0000
Mean Absolute Percentage Error: 0.0000
Median Absolute Error: 0.0000
Explained Variance Score: 1.0000
Coefficient of determination R^2: 1.0000
Mean Squared Error: 0.0000
Mean Absolute Error: 0.0000
Mean Absolute Percentage Error: 0.0000
Median Absolute Error: 0.0000
Explained Variance Score: 1.0000
Coefficient of determination R^2: 1.0000
```

PERFORMANCE OF TEST DATA

```
Root Mean Squared Error: 0.0000
Mean Absolute Error: 0.0000
Mean Absolute Percentage Error: 0.0000
Median Absolute Error: 0.0000
Explained Variance Score: 1.0000
Coefficient of determination R^2: 1.0000
Mean Squared Error: 0.0000
Mean Absolute Error: 0.0000
Mean Absolute Percentage Error: 0.0000
Median Absolute Error: 0.0000
Explained Variance Score: 1.0000
Coefficient of determination R^2: 1.0000
```

CHECKING FOR ASSUMPTIONS

NORMALITY

```
plt.hist(Errors_Train)
```



```
plt.hist(Errors_Test)
```



```
sns.kdeplot(Errors_Train, shade=True)
```



```
sns.kdeplot(Errors_Test, shade=True)
```



```
plt.boxplot(Errors_Train)
```



```
plt.boxplot(Errors_Test)
```



```
Errors_Train = np.sort(Errors_Train)
Errors_Test = np.sort(Errors_Test)
```

```
plt.plot(Errors_Train,[i/Errors_Train.shape[0]
                       for i in range(Errors_Train.shape[0])])
plt.plot(Errors_Test,[i/Errors_Test.shape[0]
                      for i in range(Errors_Test.shape[0])])
```



```
plt.hist(Errors_Train,cumulative=True,bins=Errors_Train.shape[0])
plt.hist(Errors_Test,cumulative=True,bins=Errors_Test.shape[0])
```


HOMOSCEDASTICITY

```
plt.figure(figure=None, dpi=None, facecolor='w', edgecolor='k')
plt.scatter(y_train,y_train-y_pred,c='b',s=40,alpha=0.5)
plt.scatter(y_test,y_test-y_test_pred,c='g',s=40,alpha=0.5)
plt.hlines(y=0,xmin=0,xmax=50)
plt.title('Residual Plot using training (blue) and test (green) data')
plt.ylabel('Residuals')
```



```
plt.scatter(y_train,y_pred,c='b',s=40,alpha=0.5)
plt.scatter(y_test,y_test_pred,c='g',s=40,alpha=0.5)
plt.hlines(y=0,xmin=0,xmax=50)
plt.title('Residual Plot using training (blue) and test (green) data')
plt.ylabel('Residuals')
```



```
plt.scatter(y_train,y_train-y_pred,c='b',s=40,alpha=0.5)
plt.scatter(y_test,y_test-y_test_pred,c='g',s=40,alpha=0.5)
plt.hlines(y=0,xmin=0,xmax=50)
plt.title('Residual Plot using training (blue) and test (green) data')
plt.ylabel('Residuals')
```

FINAL PREDICTION DATA

```
In [1]: data_predictions = ['Test_Data1', 'ep_wsp', 'Test_2', 'Transformed Data1', 'ep_wsp', 'Predicting_Test_DATA']
```

```
Out[1]:
                    Id  Predicted
0  1460  169103.0000
1  1461  181562.7500
2  1462  189141.8750
3  1463  189420.6250
4  1464  192617.5000

                    Id  Predicted
0  1460  169103.0000
1  1461  181562.7500
2  1462  189141.8750
3  1463  189420.6250
4  1464  192617.5000
```

COMPARING MODELS

```
In [2]: RF_Model = [ RF_Model_Score, RF_CV_Score ]
        Decision_RF_Model = [ Decision_RF_Model_Score, Decision_RF_CV_Score ]
        Decision_RF_Model_1 = [ Decision_RF_Model_Score, Decision_RF_CV_Score ]
        Decision_RF_Model_2 = [ Decision_RF_Model_Score, Decision_RF_CV_Score ]
        Decision_RF_Model_3 = [ Decision_RF_Model_Score, Decision_RF_CV_Score ]
        Decision_RF_Model_4 = [ Decision_RF_Model_Score, Decision_RF_CV_Score ]
        Decision_RF_Model_5 = [ Decision_RF_Model_Score, Decision_RF_CV_Score ]
        Decision_RF_Model_6 = [ Decision_RF_Model_Score, Decision_RF_CV_Score ]
```