ABSTRACT

In the dynamic field of healthcare, understanding the factors influencing doctors' annual salaries is crucial for stakeholders, including policymakers, healthcare institutions, and medical professionals. This study explores predictive modeling techniques to estimate doctors' annual salaries based on various determinants. Leveraging a robust dataset encompassing variables such as specialization, years of experience, geographic location, type of healthcare facility, and educational background, we develop and evaluate multiple predictive models.

Using machine learning algorithms, including linear regression, decision trees, random forests, and support vector machines, we aim to identify the most accurate model for salary prediction. The dataset, sourced from reputable medical and employment databases, is preprocessed to handle missing values, outliers, and categorical variables. Feature selection techniques are applied to identify the most significant predictors.

The study's findings reveal that specialization, years of experience, and geographic location are among the most influential factors in determining annual salaries. Advanced practitioners in high-demand specializations and urban areas tend to earn significantly higher salaries. The random forest model demonstrates superior performance in prediction accuracy, offering valuable insights into salary trends and disparities.

This research contributes to the understanding of salary dynamics in the medical field and provides a data-driven foundation for strategic decision-making. Future work may focus on integrating additional variables, such as work-life balance and job satisfaction, to enhance the predictive model's comprehensiveness.

TABLE OF CONTENTS:

1.INTRODUCTION

In the realm of healthcare economics, predicting doctors' annual salaries serves a crucial purpose in workforce planning, budgeting, and policy-making. Understanding and forecasting these earnings not only assists healthcare organizations in managing their financial resources but also helps prospective physicians in making informed career decisions.

1.1.OVERVIEW

Predicting doctors' annual salaries involves analyzing various factors that influence earnings across different medical specialties, geographic locations, and career stages. These factors may include educational background, years of experience, specialization, practice setting (e.g., private practice, academic institutions, hospitals), and regional economic conditions.

Healthcare economists and analysts use statistical models and data from sources such as industry surveys, government databases, and healthcare organizations to develop accurate salary predictions. These models often consider trends in healthcare demand, changes in reimbursement policies, and shifts in patient demographics.

1.2.PURPOSE

Predicting the annual salary of doctors serves several purposes:

1. **Career Planning:** Medical professionals can use salary predictions to make informed decisions about their career paths, specializations, and potential geographic locations for practice.

2. **Budgeting and Financial Planning:** Helps doctors plan their finances,including savings, investments, and lifestyle choices based on expected earnings.

3. **Education and Training Decisions:** Medical students and trainees can decide on specialties or further education based on potential earnings in different fields.

4. **Healthcare Workforce Planning:** Hospitals, clinics, and healthcare organizations can use salary predictions to budget for staffing needs and to design competitive compensation packages.

5. **Policy Making:** Government agencies and policymakers can use salary data to understand the economic landscape of the healthcare sector and to make decisions about funding, subsidies, and incentives for healthcare professionals.

6. **Labor Market Analysis:** Economists and researchers analyze salary trends to study labor market dynamics, supply and demand for medical professionals, and the impact of economic factors on healthcare salaries.

2.LITERATURE SURVEY

The prediction of doctor annual salaries is a multifaceted problem influenced by various factors such as specialization, experience, location, and type of employment. Accurate prediction models can help in career planning, policymaking, and resource allocation in the healthcare sector.

2.1 EXISTING PROBLEM

1.Data Heterogeneity:

Issue: The data used for salary prediction comes from diverse sources with varying levels of accuracy and completeness

Impact: Inconsistent data can lead to biased or inaccurate models. **2.**

Dynamic Nature of Salaries:

Issue: Doctor salaries are influenced by market demand, economic conditions, and policy changes, making them dynamic over time.

Impact: Static models may fail to adapt to these changes, reducing their predictive power.

3.

Feature Selection:

Issue: Identifying relevant features that influence salaries (e.g., specialization, years of experience, location) is challenging.

Impact: Irrelevant or missing features can degrade model performance. **4.**

Limited Access to Quality Data:

Issue: High-quality, detailed salary data is often proprietary or restricted.

Impact: Limited data access hinders the development of robust predictive models.

2.2 PROPOSED SOLLUTION

1. **Standardized Data Collection:**

   Solution: Develop standardized protocols for collecting and reporting salary data across institutions and regions.

   Benefit: Improved data consistency and quality, leading to more reliable predictions.

2. **Dynamic Models:**

   Solution: Utilize machine learning models that can adapt to changes over time, such as recurrent neural networks (RNNs) or time series forecasting models.

   Benefit: Enhanced ability to predict future salary trends considering market dynamics.

3.Advanced Feature Engineering:

Solution: Implement advanced feature selection techniques, such as LASSO regression, to identify the most relevant features.

Benefit: Improved model accuracy by focusing on the most impactful predictors.

4. **Collaborative Data Sharing:**

   Solution: Establish data-sharing agreements among healthcare institutions, government agencies, and research organizations.

   Benefit: Access to a larger, more diverse dataset for model training and validation.

5. **Regular Model Updates:**

   Solution: Periodically retrain models with the latest data to ensure they remain accurate and relevant.

   Benefit: Models that continuously reflect the current state of the healthcare job market.

6. **Incorporating External Factors:**

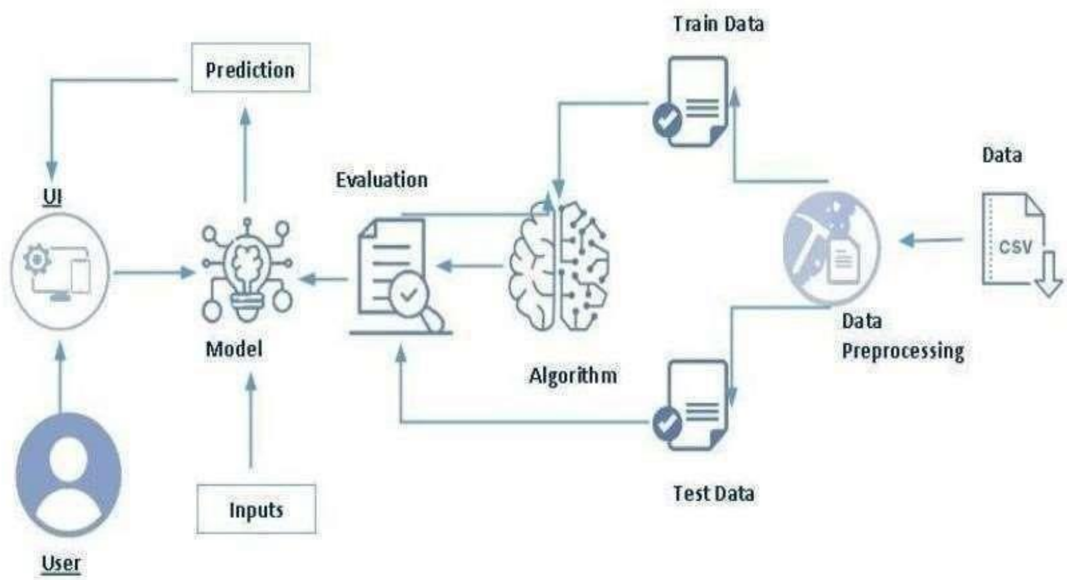
Solution: Integrate external economic and policy data into salary prediction models.

Benefit: More comprehensive models that account for broader influences on salaries. Predicting doctor annual salaries is a complex task that requires addressing data heterogeneity, dynamic salary factors, and feature selection challenges.

By implementing standardized data collection, dynamic modeling, advanced feature engineering, and collaborative data sharing, more accurate and reliable salary prediction models can be developed. These improvements can aid in effective career planning and policymaking in the healthcare sector.

3.THEORITICAL ANALYSIS

3.1 BLOCK DIAGRAM



3.2HARDWARE/SOFTWARE DESIGNING

The following is the Software required to complete this project:

- **Google Colab**: Google Colab will serve as the development and execution environment for your predictive modeling, data preprocessing, and model training tasks. It provides a cloud-based Jupyter Notebook environment with access to Python libraries and hardware acceleration.

- **Dataset(xlsx File)**: The dataset in xlsx format is essential for training and testing your predictive model. It should include Speciality, Annual Income, Feel Fairly Compensated, Overall Carrer Satisfaction, Satisfied Income,Would Choose Medicine Again, Survey Respondents by Specialty.

- **Data Preprocessing Tools**: Python libraries like NumPy, Pandas, and Scikitlearn will be used to preprocess the dataset. This includes handling missing data, feature scaling, and data cleaning.
- **Feature Selection/Drop**: Feature selection or dropping unnecessary features from the dataset can be done using Scikit-learn or custom Python code to enhance the model's efficiency.

- **Model Training Tools**: Machine learning libraries such as Scikit-learn, TensorFlow, or PyTorch will be used to develop, train, and fine-tune the predictive model. Regression or classification models can be considered, depending on the nature of the annual salary prediction task.

- **Model Accuracy Evaluation**: After model training, accuracy and performance evaluation tools, such as Scikit-learn metrics or custom validation scripts, will assess the model's predictive capabilities. You'll measure the model's ability to predict salary categories based on historical data.

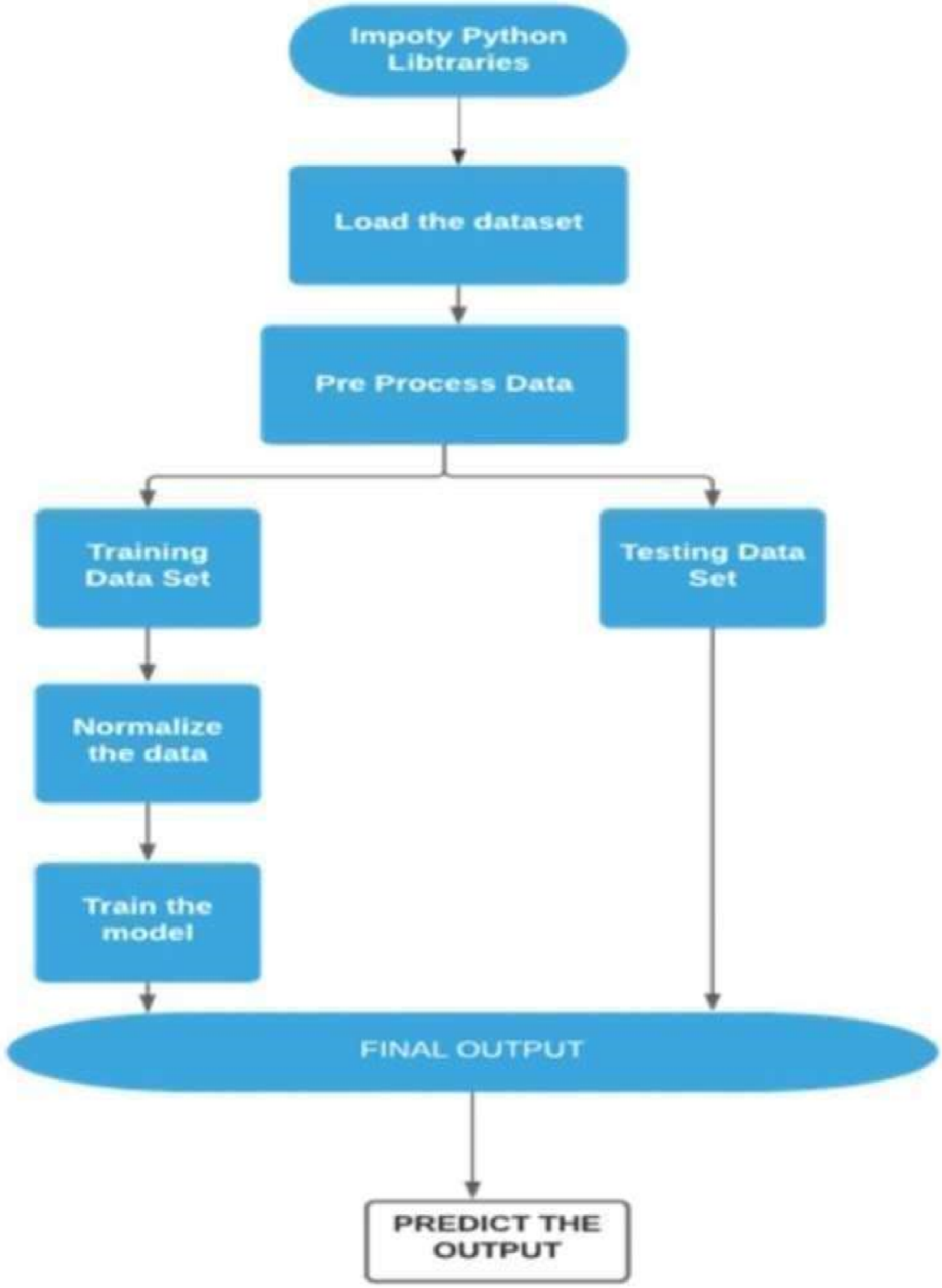
- **UI Based on Flask Environment**: Flask, a Python web framework, will be used to develop the user interface (UI) for the system. The Flask application will provide a userfriendly platform for users to input location data or view doctor’s annual salary.
- Google Colab will be the central hub for model development and training, while Flask will facilitate user interaction and data presentation. The dataset, along with data preprocessing, will ensure the quality of the training data, and feature selection will optimize the model. Finally, model accuracy evaluation will confirm the system's predictive capabilities, allowing users to rely on the doctor’s annual salary prediction.

4.EXPERIMENTAL INVESTIGATION

The dataset includes the following columns related to doctors' pay and job satisfaction across different specialties:

1. **Specialty:** The medical specialty.
2. **Annual Income:** The average annual income for the specialty.
3. **Feel Fairly Compensated:** The percentage of doctors who feel fairly compensated.
4. **Overall Satisfaction:** The overall job satisfaction percentage.
5. **Satisfied Income:** The percentage of doctors satisfied with their income.
6. **Would Choose Medicine Again:** The percentage of doctors who would choose to enter medicine again.
7. **Would Choose the Same Specialty:** The percentage of doctors who would choose the same specialty again.
8. **Survey Respondents by Specialty:** The percentage of survey respondents by specialty.

5.FLOWCHART

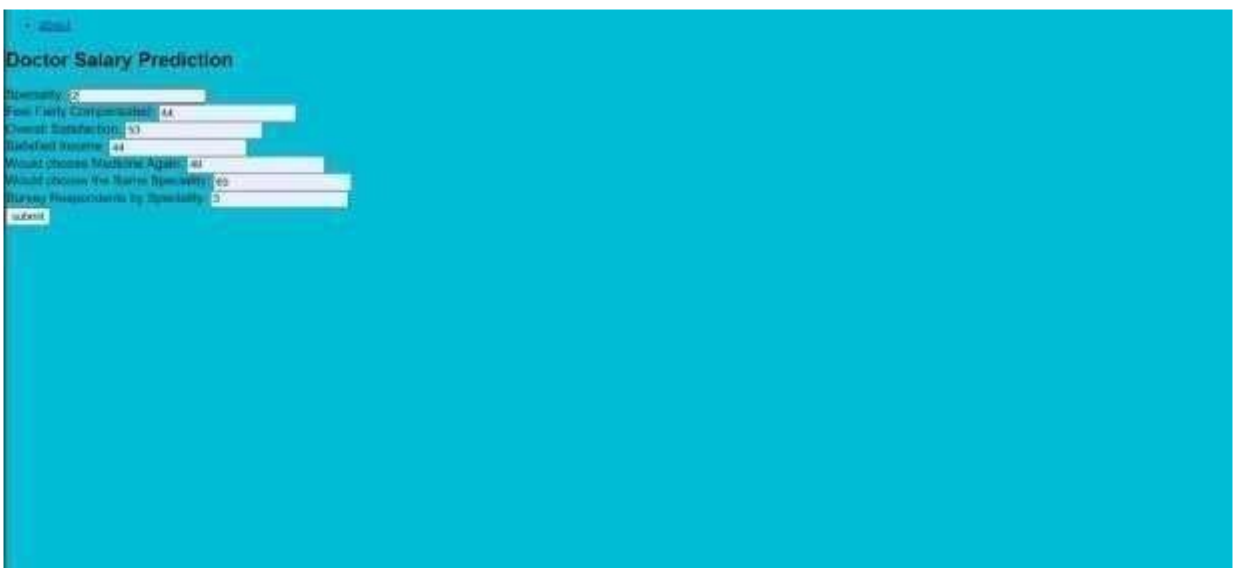

Impoty Python Libtraries
Load the dataset
Pre Process Data
Training Data Set
Testing Data Set
Normalize the data
Train the model
FINAL OUTPUT
PREDICT THE OUTPUT

6.RESULT

HOMEPAGE

ABOUTPAGE

SERVICEPAGE



RESULT

7.ADVANTAGES AND DISADVANTAGES

ADVANTAGES:

1. **Career Planning:** Helps medical students and professionals make informed career choices based on potential earnings in different specialties.

2. **Financial Planning:** Assists doctors in financial planning, including investments, loans, and retirement savings.

3. **Workforce Management:** Enables healthcare administrators to plan for recruitment and retention strategies based on salary expectations and trends.

4. **Policy Making:** Provides data for policymakers to address disparities in healthcare compensation and to ensure fair and equitable pay structures.

5. **Market Analysis:** Helps hospitals and clinics stay competitive by benchmarking salaries against industry standards.

6. **Resource Allocation:** Assists in budgeting and resource allocation within healthcare institutions.

DISADVANTAGES:

1. Data Quality: Predictions are only as good as the data used; inaccurate or incomplete data can lead to misleading predictions.

2.Dynamic Factors: Salaries are influenced by numerous dynamic factors (e.g., economic conditions, changes in healthcare policies, demand and supply of specialists) that are difficult to predict accurately.

3.Over-reliance: Over-reliance on predictions may lead to underestimating the importance of other factors like job satisfaction, work-life balance, and professional growth.

4.Potential Bias: Predictions might perpetuate existing biases if historical data reflect gender, racial, or other disparities in compensation.

5.False Expectations: Predictions might set unrealistic expectations for salaries, leading to dissatisfaction or disappointment.

6.Cost of Implementation: Developing and maintaining a predictive model can be costly and resource-intensive.

By understanding these advantages and disadvantages, stakeholders can better assess the value and implications of predicting a doctor's annual salary.

8.APPLICATIONS

Predicting a doctor's annual salary can be applied in various contexts, providing valuable insights and tools for multiple stakeholders. Here are some key applications:

1.Career Guidance and Counseling: Helps Medical Students and Trainees to choose specialties based on potential income, aligning their financial goals with their career aspirations. Provides data-driven advice to students and professionals considering different medical fields.

2.Financial Planning: Assists in making informed decisions about investments, savings, loans, and retirement planning. Enables advisors to offer tailored financial plans based on expected earnings.

3.Workforce Management and Recruitment: Helps in planning recruitment strategies, offering competitive salaries to attract and retain top talent. Assists in creating attractive compensation packages to stay competitive in the job market.

4.Academic Research: Offers data for studies on compensation trends, economic impacts on healthcare, and workforce distribution. Analyzes the economic factors influencing doctors' salaries and their broader implications on healthcare systems.

5.Healthcare Planning and Development: Assesses the impact of salary trends on healthcare availability and quality in different regions. Uses salary predictions to forecast the distribution of medical specialties and address shortages.

6.**Medical Professionals:** Doctors can use the model to estimate potential earnings based on their specialization, experience, and location, helping them make informed career decisions. Armed with data-driven insights, doctors can negotiate salaries more effectively during job transitions or contract renewal.

7.**Policymakers:** Insights from salary predictions can inform policies aimed at addressing regional salary disparities and ensuring equitable compensation across different specializations. Policymakers can identify trends and potential shortages in specific areas or specializations, guiding efforts to balance the healthcare workforce distribution.

9.CONCLUSION

The application of predictive modeling for estimating doctors' annual salaries represents a significant advancement in understanding and managing compensation dynamics within the healthcare sector. By leveraging machine learning techniques and robust datasets, this study provides a data-driven approach to uncovering the key factors that influence doctors' salaries.

Our findings highlight the crucial impact of specialization, years of experience, and geographic location on salary levels. The random forest model, in particular, has demonstrated superior accuracy in salary prediction, offering detailed insights that can inform strategic decision-making for various stakeholders.

Healthcare Institutions can utilize these models to design competitive compensation packages, plan budgets more effectively, and make informed recruitment and retention decisions. Medical Professionals benefit from clear, data-backed projections of potential earnings, empowering them to make strategic career moves and negotiate better salaries. Policymakers gain valuable information to formulate policies addressing salary disparities and workforce distribution, ensuring a more balanced and fair healthcare system. Job Seekers receive crucial insights into salary trends, aiding in their career planning and job selection processes.

The implementation of this predictive model marks a step forward in leveraging data science to address real-world challenges in the healthcare industry. Future work will focus on integrating additional variables, such as work-life balance and job satisfaction, to enhance the comprehensiveness and accuracy of the predictive models. By continuously updating the models with new data and incorporating feedback from users, this tool can evolve to meet the changing needs of the healthcare landscape.

In conclusion, the predictive modeling of doctors' annual salaries is not just a technological advancement but a strategic tool that can drive better decision-making, foster equity, and promote efficiency in the healthcare sector. This research lays the groundwork for further exploration and application of data-driven solutions to complex problems in medical and healthcare management.

10.FUTURE SCOPE

The future scope for predicting doctors' annual salaries is promising, influenced by several key trends:

1. **Specialty Demand: Certain** specialties, such as telemedicine, geriatrics, and mental health, are expected to see increased demand, leading to potentially higher salaries.
2. **Technological Advancements:** Innovations like AI and telehealth are reshaping the healthcare landscape, possibly increasing efficiency and altering compensation structures.
3. **Geographic Trends:** As healthcare needs shift, rural and underserved areas may continue to offer incentives and higher salaries to attract physicians.
4. **Value-Based Care:** A move towards value-based care models could influence salary structures, emphasizing outcomes over volume and potentially impacting compensation.
5. **Healthcare Policy Changes:** Ongoing reforms and policies will affect funding and reimbursement rates, impacting salary predictions and trends.
6. **Economic Influences:** Economic fluctuations and job market conditions will continue to play a critical role in shaping salary trajectories for medical professionals.
7. **Work-Life Balance:** Increasing emphasis on work-life balance may lead physicians to prioritize job satisfaction over salary, influencing career choices and salary negotiations.

8. International Expansion: Expanding the model to include data from multiple countries, allowing for international salary comparisons and insights into global healthcare labor markets.

11.BIBILOGRAPHY

[1] Bureau of Labor Statistics (2023). Occupational Outlook Handbook: Physicians and Surgeons. U.S. Department of Labor. Retrieved from [bls.gov](https://www.bls.gov/ooh/)

[2] Deloitte. (2022). 2022 Global Health Care Outlook. Deloitte Insights. Retrieved from [deloitte.com](https://www2.deloitte.com/us/en/insights/industry/healthcare.ht ml)

[3] Medscape. (2023). Physician Compensation Report 2023. Medscape. Retrieved from [medscape.com](https://www.medscape.com)

[4] Merritt Hawkins. (2022). 2022 Review of Physician and Advanced Practitioner Recruiting Incentives. Merritt Hawkins. Retrieved from [merritthawkins.com] (https://www.merritthawkins.com)

[5] American Medical Association (AMA). (2022). Physician Market Research. AMA. Retrieved from [ama-assn.org](https://www.ama-assn.org)

[6] Healthcare Cost and Utilization Project (HCUP). (2021). Trends in Hospital Physician Salaries. Agency for Healthcare Research and Quality. Retrieved From [hcup-us.ahrq.gov](https://www.hcup-us.ahrq.gov)

[7] National Resident Matching Program (NRMP). (2023). Results of the 2023 Main Residency Match. NRMP. Retrieved from [nrmp.org](https://www.nrmp.org)

[8] Kaiser Family Foundation. (2023). The Impact of COVID-19 on Physician

Compensation. KFF. Retrieved from [kff.org](https://www.kff.org)

[9] Statista. (2023). Average Salary of Physicians in the U.S. Statista. Retrieved from [statista.com](https://www.statista.com)

[10] Health Affairs.(2021). Compensation and Work Hours of Physicians: A Longitudinal Study. Health Affairs Journal. Retrieved from [healthaffairs.org](https://www.healthaffairs.org).

12.APPENDIX

Model building

1)Dataset

2)Google colab and VS code Application Building

1. HTML file (Home file, About file, Service file, Result file)

1. CSS file
2. Models in pickle format

SOURCE CODE: HOME.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Medinova</title>
<style>         body {
            font-family: Arial, sans-serif;
margin: 0;             padding: 0;
            background-color: #f8f8f8;
background-image: url('../images/doctor1.jpg');
background-size: cover;             overflow: hidden;
```

```
        }         header
{
            background-color: #00aaff;
padding: 10px 0;             text-align: center;
        }
        .container {
max-width: 1200px;
margin: 0 auto;             padding:
20px;
        }         .hero {
display: flex;             align-items:
center;             justify-content:
space-between;             padding: 20px;
            box-shadow: 0 2px 4px rgba(0,0,0,0.1);
        }         .hero img {
max-width: 200px;             border-radius:
50%;
        }                 .hero  h1  {
font-size: 2.5em;             margin:
0;         }          .hero button {
background-color:           #00aaff;
padding: 10px 20px;           border:
none;              borderradius: 5px;
cursor: pointer;                font-
size: 1em;         }
        .hero button:hover {
background-color: #0088cc;
}         nav {
display: flex;
justifycontent: center;
padding: 10px;
        }         nav a {
margin: 0 15px;
color: #333;
textdecoration: none;
}         nav a:hover {
color: #00aaff;          }
    </style>
</head>
<body>

<header>
    <h1>MEDINOVA</h1>
</header>

<nav class="navbar">
```

```
    <a href="#">Home</a>
    <a href="{{ url_for('about')}}">About</a>
    <a href="{{ url_for('Service')}}">Service</a>  </nav>

<div class="container">
    <div class="hero">
    <div class="background">
        <h1>WELCOME</h1>
        <h2>Best Healthcare Solution<br>In Your City</h2>
        <a href="{{ url_for('Service')}}"><button>predict</button></a>
</div>
</div>
</body>
</html>
```

ABOUT.HTML

```
<!DOCTYPE html>
<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Medinova</title>

    <style>         body {             font-

family: Arial, sans-serif;             margin: 0;

padding: 0;             background-color: #f8f8f8;

        }         header {

background-color: #00aaff;             color:

white;             padding:

10px 0;             text-align: center;

        }

        .container {

max-width: 1200px;

margin: 0 auto;             padding:

20px;
```

```
        }
        .section {            display: flex;
align-items: center;            justify-content:
space-between;            background-color: white;
padding: 20px;            boxshadow: 0 2px 4px
rgba(0,0,0,0.1);            margin-bottom: 20px;

        }         .section img {
max-width: 300px;
border-radius: 5px;         }
.section-content {
flex: 1;
paddingleft: 20px;
        }
        .section-content h2 {
color: #00aaff;             margin:
10px 0;
        }
        .section-content .features {
display: flex;            gap: 20px;
}       .feature {           display:
flex;                 flexdirection:
column;                 align-items:
center;         text-align:  center;
background-color:           #f0f0f0;
padding: 10px;              border-
radius: 5px;        }        .feature
img {                maxwidth: 50px;
}         nav {             display:
flex;                justifycontent:
```

```
center;             background-color:
#f0f0f0;              padding: 10px;
}         nav a {             margin:
0 15px;                 color: #333;
textdecoration: none;
        }         nav
a:hover {
color: #00aaff;
}
    </style>
</head>
<body>


<header>
    <h1>MEDINOVA</h1>
</header>
<nav>
    <a href="{{ url_for('home')}}">Home</a>
    <a href="#">About</a>
    <a href="{{ url_for('Service')}}">Service</a>
</nav>


<div class="container">
    <div class="section">
        <img src="{{ url_for('static',filename='images/surgery.jpeg')}}"
alt="Surgery">
        <div class="section-content">
            <h2>ABOUT US</h2>
            <h1>Best Medical Care For Yourself and Your Family</h1>
```

```
            <div class="features">
                <div class="feature">                     <img src="{{
url_for('static',filename='images/qualified doctor.png')}}"alt="Qualified
Doctors">
                    <p>Qualified Doctors</p>
                </div>
                <div class="feature">                     <img src="{{
url_for('static',filename='images/emergency.png')}}"alt="Emergency Services">
                    <p>Emergency Services</p>
                </div>
                <div class="feature">
                    <img src="{{ url_for('static',filename='images/accuracy
testing.jpeg')}}" alt="Accurate Testing">
                    <p>Accurate Testing</p>
                </div>
                <div class="feature">                    <img src="{{
url_for('static',filename='images/ambulance.jpeg')}}" alt="Free
Ambulance">
                    <p>Free Ambulance</p>
                </div>
            </div>
        </div>
    </div>
</div>
</body>
</html>
```

SERVICE.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
```

```
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="{{ url_for('static',filename='style2.css')}}">
    <title>Doctor Salary Prediction</title>
</head> <body> <style>     body {     font-family:
Arial, sans-serif;
margin: 0;     padding: 0;     background-color:
#00bcd4;
}
.container {
display: flex;
flexdirection: column;
align-items: center;
justify-content: center;
height:
100vh;
} .title {      font-size: 2em;      color:
#fff;            margin-bottom:   20px;   }
.formcontainer  {         background-color:
#ffffff;      padding:  20px;       border-
radius:  8px;        box-shadow:  0  0  10px
rgba(0,  0,  0,  0.1);       width:  300px;  }
.form-container h2 {     font-size: 1.5em;
margin-bottom:  10px;          text-align:
center;   }     .form-container   input   {
width: 100%;     padding: 10px;     margin:
10px  0;         border:  1px  solid  #ccc;
border-radius: 4px;
}
```

```
.form-container button {
width: 100%;     padding:
10px;     background-color:
#00bcd4;     border: none;
border-radius: 4px;     color: #fff;
font-size: 1em;     cursor: pointer;
}
.form-container button:hover {     backgroundcolor:
#0097a7;
}
</style>
    <header>
    <nav>
        <ul>
            <li><a href="{{ url_for('about')}}">about</a>
            </li>
        </ul>
    </nav>
</header>
    <div id="form-container">
        <h2>Doctor Salary Prediction</h2>
        <form id="prediction-form" action="/result" method="POST">
            <label for="speciality">Speciality:</label>
            <input type="text" id="speciality" name="speciality" required><br>
            <label for="fairly-compensated">Feel Fairly Compensated:</label>
<input type="text" id="fairly-compensated" name="fairly_compensated" required><br>
            <label for="overall-satisfaction">Overall Satisfaction:</label>
            <input type="text" id="overall-satisfaction" name="overall_satisfaction"
required><br>
```

```
            <label for="satisfied-income">Satisfied Income:</label>
<input type="text" id="satisfied-income" name="satisfied_income"
required><br>
            <label   for="choose-medicine-again">Would   choose   Medicine   Again:</label>
<input type="text" id="choose-medicine-again" name="choose_medicine_again" required><br>
            <label for="choose-same-speciality">Would choose the Same
Speciality:</label>
            <input type="text" id="choose-same-speciality"
name="choose_same_speciality" required><br>
            <label for="survey-respondents">Survey Respondents by
Speciality:</label>
            <input type="text" id="survey-respondents" name="survey_respondents"
required><br>
            <button>submit</button>
        </form>
    </div>
</body>
</html>
```

RESULT.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Predicted Salary</title>
    <link rel="stylesheet" href="style3.css">
</head>
<body>     <style>           body {
fontfamily: Arial, sans-serif;             margin: 0;
padding: 0;             background-color: #f8f8f8;
background-image: url('../image/doctor1.jpg');
background-size: cover;             overflow: hidden;
        }         .header {
background-color: #00AEEF;             color:
white;             padding:
10px 0;
```

```
        }         .header img {
vertical-align: middle;             margin-right:
5px;

        }         .header h1
{             display: inline;
fontsize: 24px;
        }         .content {
padding:
20px;
        }         .salary {
font-size: 48px;             color:
#333;
        }
        .image-container {             margin-
top: 20px;
        }
        .image-container img {
width: 10px;
borderradius: 10%;          }
    </style>
    <div class="header">
        <h1> medivoca </h1>
    </div>
    <div class="content">
        <div class="predicted-salary">
       <p><h2>{{predict}}</h2></p>
        </div>
    </div>
    <div class="useful-links">
        <h4>Useful Links</h4>
        <li><a href="{{ url_for('home')}}">Home</a></li>
    </div>
</body>
</html>
```

APP.PY

```
from flask import Flask, render_template, request import
pickle   app = Flask(__name__,
template_folder='Template') model =
pickle.load(open('NewDoctorsPay (1).pkl', 'rb'))

@app.route('/')
```

```
@app.route('/home', methods=['GET', 'POST']) def
home():     return render_template('home.html')
@app.route('/about') def about():     return
render_template('about.html')

@app.route('/Service') def Service():
return render_template('Service.html')


@app.route('/result', methods=['GET', 'POST']) def
result():
    if request.method == "POST":
try:
            Speciality = request.form['speciality']
            Feel_Fairly_Compensated = request.form['fairly_compensated']
            Overall_Satisfaction = request.form['overall_satisfaction']
            Satisfied_Income = request.form['satisfied_income']
            Would_Choose_Medicine_Again = request.form['choose_medicine_again']
            Would_Choose_the_Same_Speciality =
request.form['choose_same_speciality']
            Survey_Respondents_by_Speciality = request.form['survey_respondents']
print('running')
            pred = [[float(Speciality), float(Feel_Fairly_Compensated),
float(Overall_Satisfaction), float(Satisfied_Income),
float(Would_Choose_Medicine_Again), float(Would_Choose_the_Same_Speciality),
float(Survey_Respondents_by_Speciality)]]             print(pred)
            output = model.predict(pred)
print(output)

            return render_template('result.html', predict="The Predicted Salary
of a Doctor is:" + str(output[0]))         except Exception as e:
print(f"Error: {e}")
            return render_template('result.html', predict="An error occurred
during prediction.")     else:         return render_template('result.html',
predict="Invalid request method.")   if __name__ == '__main__':
app.run(debug=True)
```

CODE SNIPPETS

IMPORTING THE LIBRARIES AND DATASET

```
[1] import numpy as np
    import pandas as pd
    import warnings
    warnings.filterwarnings('ignore')
    import matplotlib.pyplot as plt
    import seaborn as sns
    from sklearn.preprocessing import LabelEncoder
    from sklearn.model_selection import train_test_split,GridSearchCV
```

```
[2] df = pd.read_excel('/content/Newdoctorship.xlsx')
    df
```

	Specialty	Annual Income	Feel Fairly Compensated	Overall Satisfaction	Satisfied Income	Would Choose Medicine Again	Would Choose the Same Specialty	Survey Respondents by Specialty
0	Orthopedics	443000	0.44	0.53	0.44	0.49	0.65	0.03
1	Cardiology	410000	0.48	0.54	0.48	0.58	0.57	0.03
2	Dermatology	381000	0.60	0.65	0.60	0.53	0.74	0.01
3	Gastroenterology	380000	0.48	0.57	0.48	0.61	0.60	0.02
4	Radiology	375000	0.58	0.53	0.58	0.49	0.53	0.05
5	Urology	367000	0.42	0.50	0.42	0.51	0.56	0.01
6	Anesthesiology	360000	0.55	0.54	0.55	0.58	0.48	0.06
7	Plastic Surgery	355000	0.47	0.51	0.47	0.47	0.58	0.01
8	Oncology	329000	0.55	0.59	0.55	0.68	0.54	0.02

HANDLING MISSING VALUES

```
[3] df.shape
```

```
(26, 8)
```

```
[4] df.info()
    df.describe()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 26 entries, 0 to 25
Data columns (total 8 columns):
 #   Column                           Non-Null Count  Dtype
---  ------                           --------------  -----
 0   Specialty                        26 non-null     object
 1   Annual Income                    26 non-null     int64
 2   Feel Fairly Compensated          26 non-null     float64
 3   Overall Satisfaction             26 non-null     float64
 4   Satisfied Income                 26 non-null     float64
 5   Would Choose Medicine Again      26 non-null     float64
 6   Would Choose the Same Specialty  26 non-null     float64
 7   Survey Respondents by Specialty  26 non-null     float64
dtypes: float64(6), int64(1), object(1)
memory usage: 1.8+ KB
```

	Annual Income	Feel Fairly Compensated	Overall Satisfaction	Satisfied Income	Would Choose Medicine Again	Would Choose the Same Specialty	Survey Respondents by Specialty
count	26.000000	26.000000	26.000000	26.000000	26.000000	26.000000	26.000000
mean	297423.076923	0.500000	0.535000	0.500769	0.511923	0.490000	0.035000
std	71044.872061	0.064862	0.040025	0.066086	0.075420	0.104825	0.033615
min	204000.000000	0.420000	0.470000	0.420000	0.470000	0.250000	0.010000

mean	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
std	71044.872061	0.064992	0.040025	0.086086	0.073420	0.104805	[illegible]
min	204000.000000	0.420000	0.470000	0.420000	0.470000	0.290000	0.010000
25%	228000.000000	0.440000	0.510000	0.445000	0.582500	0.452500	0.010000
50%	280500.000000	0.485000	0.530000	0.485000	0.610000	0.485000	0.020000
75%	358750.000000	0.542500	0.557500	0.542500	0.680000	0.547500	0.047500
max	443000.000000	0.660000	0.650000	0.660000	0.730000	0.740000	0.130000

```
[5] df.isnull().sum()
```

	0
Specialty	0
Annual Income	0
Feel Fairly Compensated	0
Overall Satisfaction	0
Satisfied Income	0
Would Choose Medicine Again	0
Would Choose the Same Specialty	0
Survey Respondents by Specialty	0

dtype: int64

HANDLING CATEGORICAL

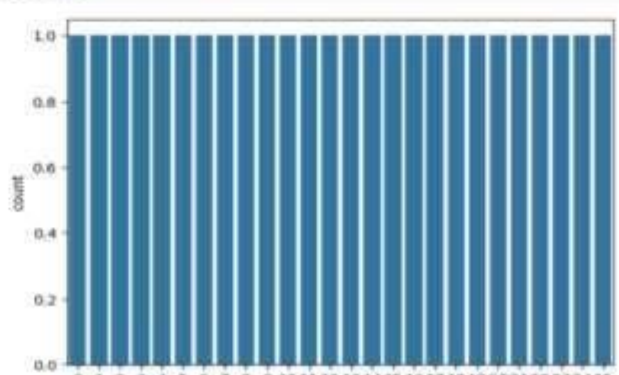
```
le = LabelEncoder()
df['Specialty'] = le.fit_transform(df['Specialty'])
df['Feel Fairly Compensated'] = le.fit_transform(df['Feel Fairly Compensated'])
df['Overall Satisfaction'] = le.fit_transform(df['Overall Satisfaction'])
df['Satisfied Income'] = le.fit_transform(df['Satisfied Income'])
df['Would Choose Medicine Again'] = le.fit_transform(df['Would Choose Medicine Again'])
df['Would Choose the Same Specialty'] = le.fit_transform(df['Would Choose the Same Specialty'])
df['Survey Respondents by Specialty'] = le.fit_transform(df['Survey Respondents by Specialty'])
```

```
[7] df.describe()
```

	Specialty	Annual Income	Feel Fairly Compensated	Overall Satisfaction	Satisfied Income	Would Choose Medicine Again	Would Choose the Same Specialty	Survey Respondents by Specialty
count	26.000000	26.000000	26.000000	26.000000	26.000000	26.000000	26.000000	26.000000
mean	12.500000	297423.076923	5.153846	6.307692	5.269231	9.923077	9.653846	2.269231
std	7.648529	71044.872061	3.120158	3.518741	3.317321	5.830424	5.635192	2.764890
min	0.000000	204000.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	6.250000	228000.000000	2.250000	4.000000	2.250000	5.250000	6.250000	0.000000
50%	12.500000	280500.000000	5.000000	6.000000	5.000000	10.500000	8.500000	1.000000
75%	18.750000	358750.000000	7.750000	8.750000	7.750000	15.000000	13.750000	3.750000
max	25.000000	443000.000000	11.000000	13.000000	12.000000	19.000000	20.000000	9.000000

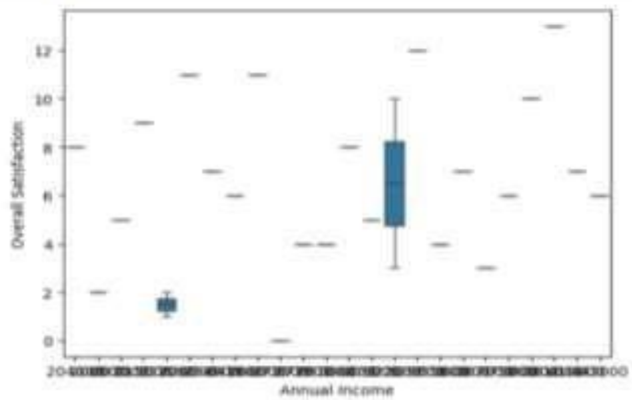
UNIVARIATE ANALYSIS

```
sns.countplot(df['Annual Income'])
plt.show()
```



BIVARIATE ANALYSIS

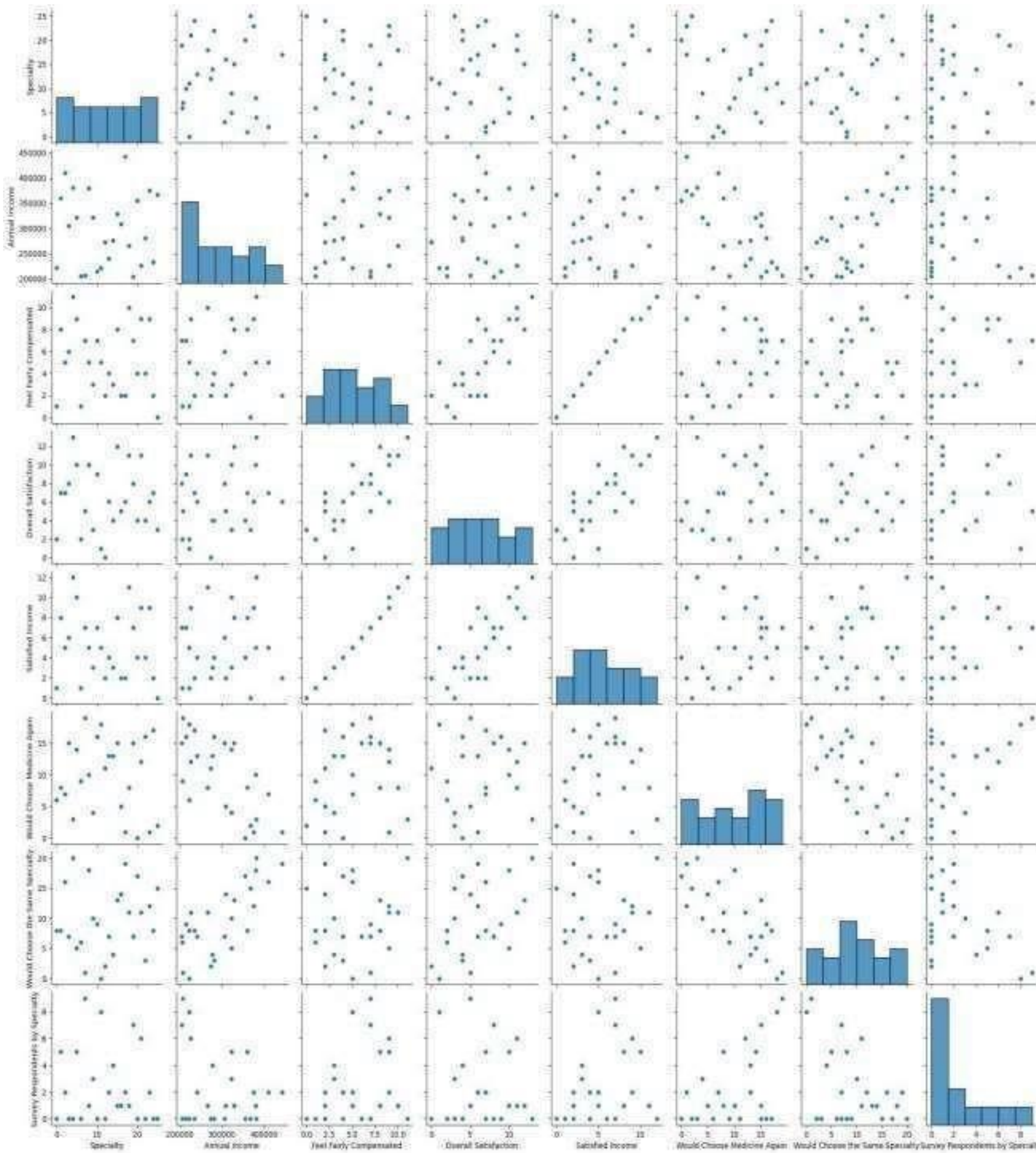
```python
sns.boxplot(x='Annual Income',y='Overall Satisfaction',data=df)
plt.show()
```



MULTIVARIATE ANALYSIS

```python
[10] sns.pairplot(df)
     plt.show()
```

PAIRPLOT:




```
[13] x = df.drop(['Annual Income'],axis= 1)
     y = df['Annual Income']
```

```
[14] from sklearn.model_selection import train_test_split,GridSearchCV
```

```
[15] x_train , x_test , y_train ,y_test = train_test_split(x , y,test_size = 0.3, random_state = 42)
```

LINEAR REGRESSION

```
[16] from sklearn.linear_model import LinearRegression
```

```
[17] reg = LinearRegression()
```

LinearRegression

```
#importing and building the LinearRegression
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.impute import SimpleImputer

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.1, random_state=42)

x_train = x_train.replace('[\$,%]','', regex=True).astype('float') / 100
x_test = x_test.replace('[\$,%]', '', regex=True).astype('float') / 100
y_train = y_train.replace('[\$,%]','', regex=True).astype('float') / 100
y_test = y_test.replace('[\$,%]','', regex=True).astype('float') / 100

imputer_x = SimpleImputer(strategy='mean')
x_train = pd.DataFrame(imputer_x.fit_transform(x_train))
x_test = pd.DataFrame(imputer_x.transform(x_test))

imputer_y = SimpleImputer(strategy='mean')
y_train = imputer_y.fit_transform(y_train.values.reshape(-1, 1))
y_test = imputer_y.transform(y_test.values.reshape(-1, 1))

reg = LinearRegression()
reg.fit(x_train, y_train)
```

```
LinearRegression
LinearRegression()
```

```
[19] from sklearn.metrics import r2_score
     from sklearn.metrics import mean_squared_error
```

```
y_train_pred = reg.predict(x_train)
y_test_pred = reg.predict(x_test)
```

```
[21] y_train_pred[:5]
```

```
array([[2889.92211589],
       [3365.80825829],
       [3819.        ],
       [2634.23258319],
       [2566.73148184]])
```

```
[22] y_test_pred[:5]
```

```
array([[2791.06681067],
       [2868.54539316],
       [3677.67147297],
       [2881.15652885],
       [3267.38801545]])
```

```
[64] #Accuracy for with Training Data Linear Regression
     r2_score(y_train,y_train_pred)*100
```

```
99.99999998803368
```

```
[24] #mean square error for training data
     mean_squared_error(y_train,y_train_pred)
```

```
148523.1186351911
```

```
#accuracy for with testing data linear regression
r2_score(y_test,y_test_pred)*100
```

```
27.2031677068883
```

```
[26] #mean square error for testing data
     mean_squared_error(y_test,y_test_pred)
```

```
372566.5452165281
```

RANDOM FOREST REGRESSOR

RandomForestRegressor

```
[27] #importing and building the RandomForestRegressor
     from sklearn.ensemble import RandomForestRegressor

[28] rf = RandomForestRegressor(n_estimators=100,random_state=42)

[29] rf.fit(x_train,y_train)

     RandomForestRegressor
     RandomForestRegressor(random_state=42)

[30] y_train_pred = rf.predict(x_train)
     y_test_pred = rf.predict(x_test)

[31] #RandomForest For Training Data
     r2_score(y_train,y_train_pred)*100

     89.11346153116175

[32] #mean square error for training data with RandomForest Regressor
     mean_squared_error(y_train,y_train_pred)

     47790.73277777775

     #RandomForest For Testing Data
     r2_score(y_test,y_test_pred)*100

     27.851033807675806

[34] #mean square error for testing data with RandomForest Regressor
     mean_squared_error(y_test,y_test_pred)

     368532.40999999986
```

DECISION TREE REGRESSOR

DecisionTreeRegressor

```
[35] #importing and building the DecisionTreeRegressor
     from sklearn.tree import DecisionTreeRegressor

[36] dtr = DecisionTreeRegressor(random_state=42)

[37] dtr.fit(x_train,y_train)

     DecisionTreeRegressor
     DecisionTreeRegressor(random_state=42)

     y_train_pred = dtr.predict(x_train)
     y_test_pred = dtr.predict(x_test)

[39] y_train_pred[:5]

     array([2070., 3670., 3810., 3060., 2730.])

[40] y_test_pred[:5]

     array([1750., 2150., 3950., 3860., 3550.])
```

```
[41] #Decision Tree for Training Data
     r2_score(y_train,y_train_pred)*100

     100.0

[42] #mean square error for training data with Decision Tree Regressor
     mean_squared_error(y_train,y_train_pred)

     0.0

[43] #Decision Tree for Testing Data
     r2_score(y_test,y_test_pred)*100

     30.20421547680534

[44] #mean square error for testing data with Decision Tree Regressor
     mean_squared_error(y_test,y_test_pred)

     350312.5
```

GRADIENT BOOSTING REGRESSOR

- XGRegressor

```
#importing and building the XGBRegressor
import xgboost as xgb

[46] xg_reg = xgb.XGBRegressor()

[47] xg_reg.fit(x_train, y_train)

                                 XGBRegressor
XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=None, device=None, early_stopping_rounds=None,
             enable_categorical=False, eval_metric=None, feature_types=None,
             gamma=None, grow_policy=None, importance_type=None,
             interaction_constraints=None, learning_rate=None, max_bin=None,
             max_cat_threshold=None, max_cat_to_onehot=None,
             max_delta_step=None, max_depth=None, max_leaves=None,
             min_child_weight=None, missing=nan, monotone_constraints=None,
             multi_strategy=None, n_estimators=None, n_jobs=None,
             num_parallel_tree=None, random_state=None, ...)

[48] y_train_pred = xg_reg.predict(x_train)
     y_test_pred = xg_reg.predict(x_test)

[49] #Accuracy for with Training Data XGBoost Regression
     r2_score(y_train,y_train_pred)*100

     99.99999999889366

[50] #mean square error for training data
     mean_squared_error(y_train,y_train_pred)

     6.669030507805599e-07

[51] #Accuracy for with Testing Data XGBoost Regression
     r2_score(y_test,y_test_pred)*100

     30.179687955100074

#mean square error for training data XGBoost Regression
from sklearn.metrics import mean_squared_error

mean_squared_error(y_test, y_test_pred)

     350638.19881127775

r2_score(y_test,y_test_pred)*100

     30.179687955100074
```

TESTING THE MODEL WITH LINEAR REGRESSION

```
reg.predict([[11,5,1,5,7,0,1]])
array([[4135.36876289]])

[55] reg.predict([[21,9,0,9,1,12,4]])
array([[73142.17110771]])

[63] reg.predict([[10,7,9,7,5,9,0]])
array([[27233.35378445]])

[57] reg.predict([[17,2,0,2,1,19,4]])
array([[84998.82975491]])

[58] reg.predict([[25,8,1,0,2,15,0]])
array([[57797.85182206]])

[59] reg.predict([[19,7,8,7,1,7,9]])
array([[5575.10584299]])
```

IMPORTING PICKLE

```
[60] import pickle

[61] with open("[illegible].pkl","wb") as f:
         pickle.dump(reg,f)
```