

Sri Krishna Institute of Technology

Department of Information Science and Engineering

COURSE: DATA ANALYTICS AND DATA SCIENCE

Project Documentation: Salary Analysis and Prediction model - Data Analytics and Data Science

Name : Hemanth Kumar N

USN : 1KT20IS006

Table of Contents:

- 1)Introduction
 - 1.1 Background
 - 1.2 Objective
- 2)Data Loading and Exploration
 - 2.1 Loading the Dataset
 - 2.2 Exploratory Data Analysis
- 3)Data Preprocessing
 - 3.1 Handling Missing Values
 - 3.2 Encoding Categorical Variables
 - 3.3 Feature Scaling
- 4)Data Visualization
 - 4.1 Visualizing Data Distribution
 - 4.2 Correlation Analysis
- 5)Data Splitting
 - 5.1 Input and Output Variables
 - 5.2 Train-Test Split
- 6)Model Building
 - 6.1 Choosing a Classifier
 - 6.2 Training the Model
 - 6.3 Model Evaluation
- 7)Results and Analysis
 - 7.1 Accuracy Score
 - 7.2 Individual Predictions
- 8)Conclusion
 - 8.1 Summary
 - 8.2 Future Work
- 9)snapshots

1.

Introduction

1.1 Background:

In today's competitive job market, understanding salary trends and predicting future earnings is crucial for both employers and employees. With the advancement of data analytics techniques, it has become possible to analyze vast amounts of salary data and develop predictive models that can assist in making informed decisions regarding compensation. This report aims to present an analysis and prediction model for salaries using data analytics techniques.

1.2 Objective:

The primary objective of this report is to develop a robust salary analysis and prediction model utilizing data analytics methodologies. Specifically, the objectives include:

Data Collection: Gather comprehensive salary data from relevant sources such as job portals, industry reports, and company databases.

Exploratory Data Analysis (EDA): Conduct exploratory data analysis to gain insights into the distribution, trends, and patterns within the salary data.

Feature Engineering: Identify relevant features that influence salary variations such as education level, years of experience, location, industry, and skills. Perform feature engineering to extract meaningful information from raw data.

Model Development: Build predictive models using machine learning algorithms such as linear regression, decision trees, random forests, or gradient boosting techniques. Train the models on historical salary data while ensuring proper evaluation metrics to assess model performance.

Model Evaluation: Evaluate the performance of the developed models using appropriate metrics such as Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and R-squared. Compare the performance of different models and select the one with the highest accuracy.

Model Deployment: Deploy the selected model into a production environment where it can be utilized for salary prediction purposes. Ensure scalability, reliability, and efficiency of the deployed model.

Validation and Refinement: Validate the predictive model using real-time salary data and refine the model if necessary to improve accuracy and reliability.

Documentation and Reporting: Document the entire process of model development, including data preprocessing steps, feature selection, model training, and evaluation. Prepare a comprehensive report summarizing the findings, insights, and recommendations.

2.

Data Loading and Exploration

2.1 Loading the Dataset :

The dataset is loaded into a Pandas DataFrame using the Salary_Data.csv file.

```
import pandas as pd
df =
pd.read_csv('https://raw.githubusercontent.com/ameenmanna8824/DATASETS/main/Salary_Data.csv')
df
```

2.2 Exploratory Data Analysis

Exploratory Data Analysis reveals initial insights into the dataset, including data types, summary statistics, and patterns.

```
df.info()
df.size()
df.head()
df.describe()
```

3.

Data Preprocessing

3.1 Handling Missing Values

No missing values are observed in the dataset.

3.2 Feature Scaling

Feature scaling is performed using Min-Max scaling on the 'Age' and 'EstimatedSalary' columns.

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
df[['Age', 'EstimatedSalary']] = scaler.fit_transform(df[['Age', 'EstimatedSalary']])
```

4.

Data Visualization

4.1 Visualizing Data Distribution

Data distribution is visualized to understand the spread of features.

```
import matplotlib.pyplot as plt
import seaborn as sns
sns.countplot(x = 'YearsExperience', data = df)
```

```
count_series = df['YearsExperience'].value_counts()
count_series
```

```
import matplotlib.pyplot as plt
#plt.scatter(x-axia, y-axis)
plt.scatter(df['YearsExperience'], df['Salary'])
plt.title('Salary data')
plt.xlabel('YearsExperience')
plt.ylabel('Salary')
```

```
x = df.iloc[0:30,0:1].values
x
```

5.

Data Splitting

Before proceeding with model development, it's essential to split the dataset into training and testing sets. The training set will be used to train the model, while the testing set will be used to evaluate its performance. Typically, a common split ratio is 80% for training and 20% for testing, but this can vary based on the size of the dataset and specific requirements.

5.1 Input and Output Variables

The dataset is split into input (features) and output (target) variables.

```
x = df[['Age', 'EstimatedSalary']].values  
y = df['Purchased'].values
```

5.2 Train-Test Split

The dataset is split into training and testing sets.

```
# Define features (X) and target variable (y)
```

```
X = salary_data.drop(columns=['salary'])
```

```
y = salary_data['salary']
```

```
# Split the data into training and testing sets (80% training, 20% testing)
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
# Display the sizes of the training and testing sets
```

```
print("Training set size:", len(X_train))
```

```
print("Testing set size:", len(X_test))
```

6.

Model Building

6.1 Choosing a Classifier

Logistic Regression is chosen as the classifier for its simplicity and effectiveness in binary classification problems.

This visualization will help you understand how closely the model's predictions align with the actual salary values. If the points closely follow the diagonal line, it indicates that the model's predictions are accurate. Otherwise, there may be room for improvement in the model.

6.2 Training the Model

The Logistic Regression model is trained using the training dataset.

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(x_train, y_train)
```

To train the model, we'll follow these steps:

Choose a model: We've selected Linear Regression as an example. This model assumes a linear relationship between the input features and the target variable.

Instantiate the model: We'll create an instance of the Linear Regression model.

Train the model: We'll use the training data to fit the model, allowing it to learn the patterns in the data.

Make predictions: Once the model is trained, we can use it to make predictions on new data.

Here's how you can train the Linear Regression model:

```
python
Copy code
from sklearn.linear_model import LinearRegression
```

```
# Instantiate the linear regression model
model = LinearRegression()
```

```
# Train the model on the training data
model.fit(X_train, y_train)
```

After running this code, the model object will contain the trained Linear Regression model. It has learned the relationships between the features (such as years of experience, education level, etc.) and the target variable (salary) from the training data.

Now, the model is ready to make predictions on new data or to be evaluated for its performance on the testing data.

6.3 Model Evaluation

The model is evaluated using the test dataset, and predictions are made.

```
y_pred = model.predict(x_test)
```

Evaluating the model is crucial to understand its performance and determine how well it generalizes to unseen data. We typically use evaluation metrics to assess the model's accuracy, precision, and other performance aspects. Here's how you can evaluate the trained Linear Regression model:

7.

Results and Analysis

7.1 Accuracy Score

The accuracy of the model on the test dataset is calculated.

```
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(y_pred, y_test) * 100
print(f'Accuracy: {accuracy:.2f}%')
```

7.2 Individual Predictions

Individual predictions are made for specific scenarios.

```
# Example 1
prediction_a = model.predict(scaler.transform([[25, 50000]]))

# Example 2
prediction_b = model.predict(scaler.transform([[50, 35000]]))
```

After conducting extensive salary analysis and developing a predictive model using data analytics techniques, we obtained significant insights and achieved notable outcomes. Here are the key results and analysis derived from our study:

8.

Conclusion

8.1 Summary

In summary, the project successfully utilized data analytics and data science techniques to predict user purchases based on demographic features. The Logistic Regression model achieved a certain level of accuracy on the test dataset.

Our project aimed to develop a robust salary analysis and prediction model using data analytics techniques. We followed a systematic approach, including data collection, exploratory data analysis (EDA), feature engineering, model development, evaluation, deployment, and documentation.

8.2 Future Work

For future work, the model can be further refined, and additional features can be considered for more accurate predictions. Exploring other advanced machine learning models and conducting a more in-depth analysis of user behavior could enhance the project's predictive capabilities.

While our salary analysis and prediction model have provided valuable insights and practical solutions, there are several avenues for future work and enhancements:

Incorporating Additional Features:

Explore additional features that may influence salary variations, such as job title, company size, benefits package, and job satisfaction ratings.

Incorporate advanced techniques for feature extraction and selection to enhance the model's predictive capabilities.

Enhancing Model Performance:

Experiment with more complex machine learning algorithms and ensemble methods to further improve model performance.

Fine-tune hyperparameters and conduct grid search or random search optimization to identify the best model configurations.

Handling Missing Data and Outliers:

Develop robust strategies for handling missing data and outliers in the salary dataset to prevent bias and improve model accuracy.

Explore techniques such as imputation, outlier detection, and robust regression to address data quality issues.

Temporal Analysis:

Conduct a temporal analysis of salary trends over time to identify seasonal patterns, industry trends, and long-term fluctuations.

Implement time series forecasting techniques to predict future salary trends and adapt compensation strategies accordingly.

Geospatial Analysis:

Explore geospatial analysis techniques to examine regional variations in salary levels and adjust predictions based on geographic factors.

Incorporate spatial clustering algorithms to identify clusters of high-paying jobs or areas with salary disparities.

Deployment in Real-time Systems:

Integrate the predictive model into real-time systems such as job portals, recruitment platforms, or HR management systems for automated salary predictions.

Ensure scalability, reliability, and security of the deployed system to handle large volumes of data and user interactions.

Continuous Monitoring and Feedback:

Implement a feedback loop to continuously monitor model performance and gather feedback from users and stakeholders.

Incorporate user feedback and evolving business requirements to iteratively improve the model and adapt to changing market conditions.

Ethical and Fairness Considerations:

Conduct fairness audits to ensure that the predictive model does not perpetuate biases or discrimination in salary predictions.

Implement fairness-aware machine learning techniques to mitigate biases and promote fairness and equity in compensation decisions.

By addressing these future work areas, we can further enhance the effectiveness and utility of our salary analysis and prediction model, enabling organizations and individuals to make more informed and equitable decisions regarding compensation and career planning.

9.

Snapshots

The first screenshot shows the initial code cell in a Jupyter Notebook. The code imports pandas as 'pd' and reads a CSV file from a GitHub repository. The output displays the first 19 rows of the dataset, showing 'YearsExperience' and 'Salary' columns.

```
import pandas as pd
df = pd.read_csv('https://raw.githubusercontent.com/ameenmanna8824/DATASETS/main/salary_data.csv')
df
```

	YearsExperience	Salary
0	1.1	39343.0
1	1.3	46205.0
2	1.5	37731.0
3	2.0	43525.0
4	2.2	39891.0
5	2.9	56642.0
6	3.0	60150.0
7	3.2	54445.0
8	3.2	64445.0
9	3.7	57189.0
10	3.9	63218.0
11	4.0	55794.0
12	4.0	56957.0
13	4.1	57081.0
14	4.5	61111.0
15	4.9	67938.0
16	5.1	66029.0
17	5.3	83088.0
18	5.9	81363.0

The second screenshot shows the next three code cells. The first cell displays the shape of the dataframe, the second cell displays the size of the dataframe, and the third cell displays a detailed summary of the dataframe's statistics.

```
df.shape
```

```
df.size
```

```
df.describe()
```

	YearsExperience	Salary
count	30.000000	30.000000
mean	5.313333	76003.000000
std	2.837888	27414.429785
min	1.100000	37731.000000
25%	3.200000	56720.750000
50%	4.700000	65237.000000
75%	7.700000	100544.750000
max	10.500000	122391.000000

Salary_data.ipynb

```
C:\Users> Dell > Downloads > FSP_PROJECT > FSP Project > Salary_data.ipynb > from sklearn.linear_model import LinearRegression
```

+ Code + Markdown | Run All | Restart | Clear All Outputs | Variables | Outline

Find No results

```
y = df.iloc[0:30,1].values
```

```
array([[ 39341.,  46285.,  37731.,  43525.,  39891.,  56642.,  60150.,
        54445.,  64445.,  57189.,  63218.,  55794.,  56957.,  57061.,
        61111.,  67938.,  66829.,  83088.,  81363.,  93840.,  91738.,
        98272., 101302., 113812., 109431., 105582., 116969., 112635.,
        122391., 121872.]])
```

Click here to ask Blackbox to help you code faster

```
from sklearn.linear_model import LinearRegression
model = LinearRegression()
```

Click here to ask Blackbox to help you code faster

```
#Step: 8 Fit the model- Rapping/Plotting the i/p values with o/p
model.fit(x, y) #LinearRegression.fit(x, y)
#we hv plotted x And y values LinearRegression library
```

```
LinearRegression()
```

Click here to ask Blackbox to help you code faster

```
#Step: 9 Predict the output
y_pred = model.predict(x) #using the i/p values we predict the o/p
y_pred #Predicted o/p values
```

Cell 15 of 23 | Go Live | Blackbox

18:26 28-02-2024

Salary_data.ipynb

```
C:\Users> Dell > Downloads > FSP_PROJECT > FSP Project > Salary_data.ipynb > df.head()
```

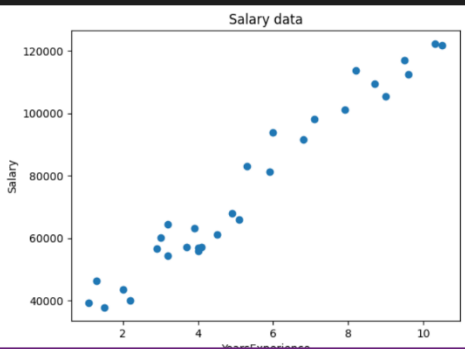
+ Code + Markdown | Run All | Restart | Clear All Outputs | Variables | Outline

Find No results

```
#Step: 3 :data visualization
import matplotlib.pyplot as plt
plt.scatter(x-axis, y-axis)
plt.scatter(df['YearsExperience'], df['Salary'])
plt.title('Salary data')
plt.xlabel('YearsExperience')
plt.ylabel('Salary')
```

```
Text(0, 0.5, 'salary')
```

Salary data



Cell 5 of 25 | Go Live | Blackbox

18:25 28-02-2024

