MENTORNESS

SALARY PREDICTION OF DATA PROFESSIONS

USING MACHINE LEARNING

Presented by MD. SHAKIB

OUR GOAL

Our mission in this internship is to construct a robust predictive model for the salaries of data professionals. This endeavor involves a structured approach encompassing several key stages:

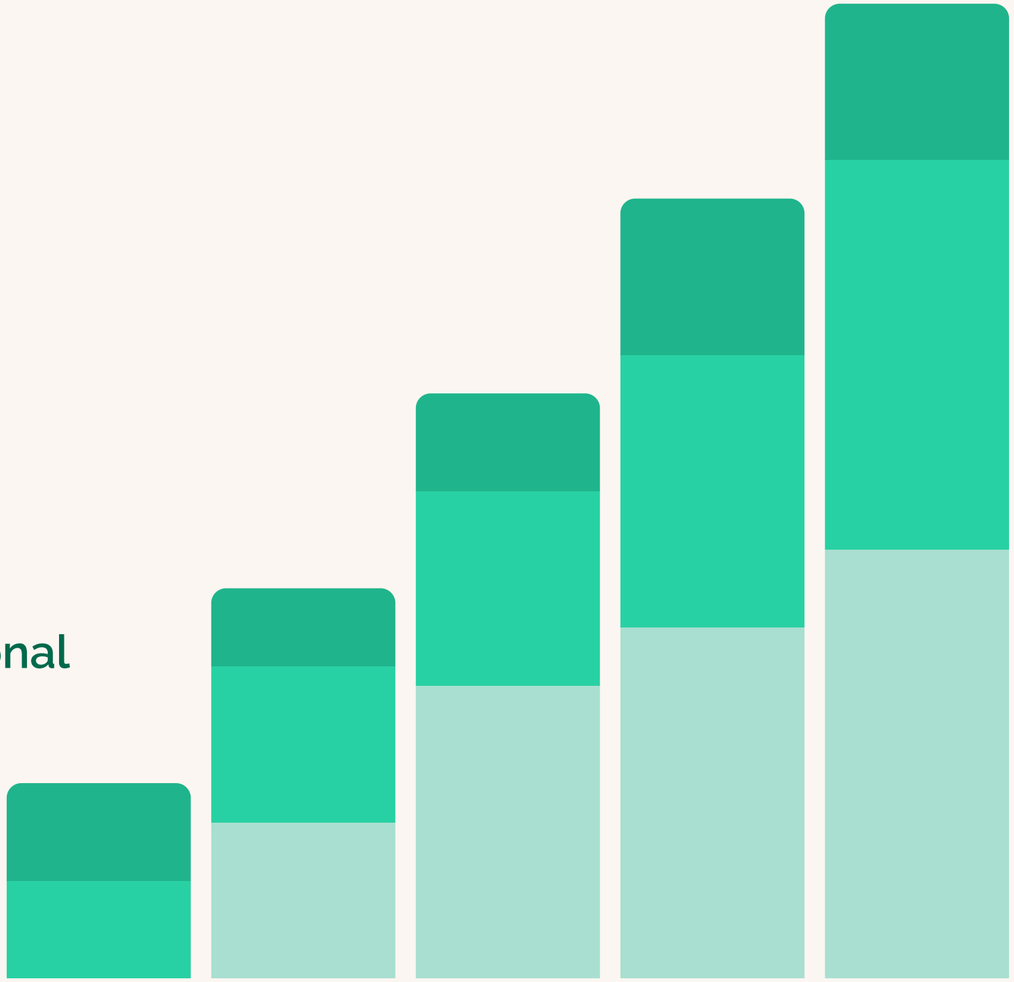
1. Exploratory Data Analysis (EDA): **Delving into the dataset to gain insights, understand patterns, and identify potential relationships between variables.**
2. Data Preprocessing: Cleaning and preparing the data for analysis, which includes handling missing values, encoding categorical variables, and scaling features.
3. Feature Engineering: Crafting new features or transforming existing ones to enhance the predictive power of the model.
4. Model Development: Building and fine-tuning machine learning models tailored to predict salary based on relevant features.
5. Model Evaluation: Assessing the performance of the developed models using appropriate metrics and validation techniques to ensure reliability and generalization.
6. ML Pipeline: Establishing an efficient and reproducible pipeline to streamline the entire process from data ingestion to model deployment, facilitating scalability and maintainability.

Through these concerted efforts, we aim to create a predictive framework that can accurately forecast the salaries of data professionals, contributing valuable insights to the field of human resources and talent management.

Dataset Description

The dataset contains the following columns:

- `FIRST NAME`: First name
- `LAST NAME`: Last name
- `SEX`: Gender
- `DOJ`: Date of joining the company
- `CURRENT DATE`: Current date of data
- `DESIGNATION`: Job role/designation
- `AGE`: Age
- `SALARY`: Target variable, the salary of the data professional
- `UNIT`: Business unit or department
- `LEAVES USED`: Number of leaves used
- `LEAVES REMAINING`: Number of leaves remaining
- `RATINGS`: Ratings or performance ratings
- `PAST EXP`: Past work experience



Data Exploratory Analysis

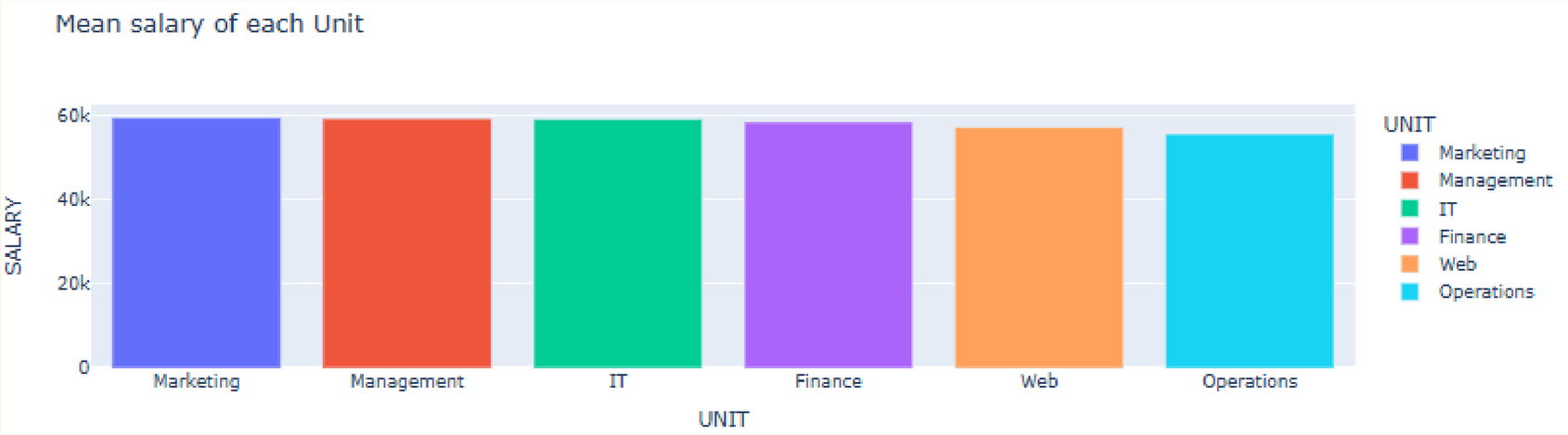
- Performing exploratory data analysis (EDA) in salary prediction involves several steps to understand the data, identify patterns, detect anomalies, and form hypotheses for further analysis.

```
[14]: import numpy as np
      import pandas as pd
      import matplotlib.pyplot as plt
      import plotly.express as px
      import seaborn as sns
      import re
      from sklearn.metrics import r2_score, mean_absolute_error, mean_squared_error
      from sklearn.model_selection import train_test_split
      from sklearn.linear_model import LinearRegression

[2]: df = pd.read_csv("Salary Prediction of Data Professions.csv")

[3]: df.head(10)
```

[3]:

	FIRST NAME	LAST NAME	SEX	DOJ	CURRENT DATE	DESIGNATION	AGE	SALARY	UNIT	LEAVES USED	LEAVES REMAINING	RATINGS	PAST EXP
0	TOMASA	ARMEN	F	5-18-2014	01-07-2016	Analyst	21.0	44570	Finance	24.0	6.0	2.0	0
1	ANNIE	NaN	F	NaN	01-07-2016	Associate	NaN	89207	Web	NaN	13.0	NaN	7
2	OLIVE	ANCY	F	7-28-2014	01-07-2016	Analyst	21.0	40955	Finance	23.0	7.0	3.0	0
				04-03-									

- Depending on the nature and amount of missing data, decide whether to drop rows/columns or impute missing values using appropriate methods (mean, median, mode, etc.).



Feature Engineering

- **Feature engineering is the process of using domain knowledge to create or transform features (variables) that enhance the performance of machine learning models.**

- Effective feature engineering can significantly improve model accuracy and performance by providing the model with more relevant and meaningful input data.

```
]: from sklearn.preprocessing import OrdinalEncoder
encoder = OrdinalEncoder()
encoded_column = encoder.fit_transform(df1[['DESIGNATION']])
df1['DESIGNATION'] = encoded_column
print(df1)
```

```
      DESIGNATION  SALARY        UNIT  LEAVES USED  LEAVES REMAINING  RATINGS  \
0             0.0   44570     Finance         24.0               6.0      2.0
2             0.0   40955     Finance         23.0               7.0      3.0
3             0.0   45550          IT         22.0               8.0      3.0
6             0.0   40339   Marketing         19.0              11.0      5.0
8             4.0   63478  Operations         20.0              10.0      3.0
...           ...     ...         ...          ...               ...      ...
2634          5.0  185977  Management         15.0              15.0      5.0
2635          0.0   45758          IT         17.0              13.0      2.0
2636          0.0   47315         Web         29.0               1.0      5.0
2637          0.0   45172         Web         23.0               7.0      3.0
2638          0.0   49176   Marketing         17.0              13.0      2.0

      PAST EXP
0            0
2            0
3            0
6            0
8            1
...        ...
2634        10
2635         0
2636         0
2637         1
2638         2

[2631 rows x 7 columns]
```

Training & Testing of Model

- **Overfitting and Underfitting:** Ensure the model is not overfitting (performing well on training data but poorly on testing data) or underfitting (performing poorly on both training and testing data).

- **Model Validation:** Compare the performance metrics from the training and testing phases to validate the model's ability to generalize to new, unseen data.

- **Error Analysis:** Analyze errors and misclassifications to identify patterns and improve the model further, if needed.

Splitting into Training & Testing

```
[45]: X_train,X_test,y_train,y_test = train_test_split(x,y,test_size = 0.2,random_state = 42)
```

```
[43]: x = df1[['DESIGNATION','UNIT','LEAVES USED','LEAVES REMAINING','RATINGS','PAST EXP']]
      y = df1[['SALARY']]
      print(x)
```

	DESIGNATION	UNIT	LEAVES USED	LEAVES REMAINING	RATINGS	PAST EXP
0	0.0	0.0	24.0	6.0	2.0	0
2	0.0	0.0	23.0	7.0	3.0	0
3	0.0	1.0	22.0	8.0	3.0	0
6	0.0	3.0	19.0	11.0	5.0	0
8	4.0	4.0	20.0	10.0	3.0	1
...	...	...	...	...	...	...
2634	5.0	2.0	15.0	15.0	5.0	10
2635	0.0	1.0	17.0	13.0	2.0	0
2636	0.0	5.0	29.0	1.0	5.0	0
2637	0.0	5.0	23.0	7.0	3.0	1
2638	0.0	3.0	17.0	13.0	2.0	2

[2631 rows x 6 columns]

```
[44]: print(y)
```

	SALARY
0	44570
2	40955
3	45550
6	40339
8	63478
...	...
2634	185977
2635	45758
2636	47315
2637	45172
2638	49176

[2631 rows x 1 columns]


```
[44]: print(y)
```

	SALARY
0	44570
2	40955
3	45550
6	40339
8	63478
...	...
2634	185977
2635	45758
2636	47315
2637	45172
2638	49176

[2631 rows x 1 columns]


```
[46]: len(X_train)
[46]: 2104
[47]: len(y_train)
[47]: 2104
[48]: len(X_test)
[48]: 527
[49]: len(y_test)
[49]: 527
```

- **2104 rows in Train**
- **527 rows in Test**

Model Development & Evaluation

1.Performance Metrics:

- **Mean Absolute Error (MAE):** Average absolute difference between predicted and actual values.
- **Mean Squared Error (MSE):** Average squared difference between predicted and actual values.
- **Compare and Select:** Compare the evaluation metrics of all models.
- **Validation:** Perform further validation using techniques like k-fold cross-validation to ensure the selected model generalizes well.
- **Deploy the Model:**Once the best model is identified, it can be deployed for making salary predictions on new data.

LINEAR REGRESSION

```
model = LinearRegression()
model.fit(X_train,y_train)
```

```
LinearRegression()
```

```
pred = model.predict(X_test)
print(X_test)
```

```
      DESIGNATION  UNIT  LEAVES USED  LEAVES REMAINING  RATINGS  PAST EXP
229           0.0   3.0         22.0               8.0      2.0         0
326           0.0   4.0         22.0               8.0      3.0         1
934           0.0   4.0         23.0               7.0      2.0         0
1363          0.0   5.0         23.0               7.0      3.0         0
1429          0.0   3.0         25.0               5.0      5.0         1
...           ...   ...          ...               ...      ...       ...
2085          5.0   1.0         18.0              12.0      2.0         9
628           0.0   1.0         15.0              15.0      3.0         0
1159          0.0   0.0         30.0               0.0      5.0         0
1228          0.0   2.0         23.0               7.0      4.0         2
1216          0.0   1.0         18.0              12.0      4.0         1

[527 rows x 6 columns]
```

```
print(pred)
```

```
[[ 39068.01674063]
 [ 49862.78466081]
 [ 39338.26648562]
 [ 39325.13645482]
 [ 49045.23838274]
 [ 93863.12255274]
 [ 59566.66740664]
 [ 59052.118608  ]
 [ 49302.66746719]
```

```
[53]: train_pred = model.predict(X_train)
      mae = mean_absolute_error(train_pred,y_train)
      print("Mean Absolute Error: ",mae)

      Mean Absolute Error:  11874.161997957106
```

```
[54]: test_pred = model.predict(X_test)
      mae = mean_absolute_error(test_pred,y_test)
      print("Mean Absolute Error: ",mae)

      Mean Absolute Error:  11663.732959004958
```

Model Development & Evaluation

- **RANDOM FOREST REGRESSOR**

RANDOM FOREST REGRESSOR

[55]:
```
from sklearn.ensemble import RandomForestRegressor
model1= RandomForestRegressor()
model1.fit(X_train,y_train)
```

```
C:\Users\shaki\AppData\Local\Temp\ipykernel_17892\1148760174.py:3: DataConversionW

A column-vector y was passed when a 1d array was expected. Please change the shape
```

[55]:
```
▾ RandomForestRegressor
RandomForestRegressor()
```

[56]:
```
y_train_pred = model1.predict(X_train)
mse_RFR = mean_absolute_error(y_train_pred,y_train)
print("Mean Absolute Error: ",mse_RFR)
```

```
Mean Absolute Error:  2154.459163215356
```

[57]:
```
y_test_pred = model1.predict(X_test)
mse_RFR1 = mean_absolute_error(y_test_pred,y_test)
print("Mean Absolute Error: ",mse_RFR1)
```

```
Mean Absolute Error:  4300.609521849778
```

- **DECISION TREE REGRESSOR**

DECISION TREE REGRESSOR

[8]:
```
from sklearn.tree import DecisionTreeRegressor,plot_tree
model2 = DecisionTreeRegressor()
model2.fit(X_train,y_train)
```

[8]:
```
▾ DecisionTreeRegressor
DecisionTreeRegressor()
```

[9]:
```
y_train_pred_tree = model2.predict(X_train)
mse_DTR = mean_absolute_error(y_train_pred_tree,y_train)
print("Mean Absolute Error: ",mse_DTR)
```

```
Mean Absolute Error:  1144.7415014484884
```

[9]:
```
y_test_pred_tree = model2.predict(X_test)
mse_DTR1 = mean_absolute_error(y_test_pred_tree,y_test)
print("Mean Absolute Error: ",mse_DTR1)
```

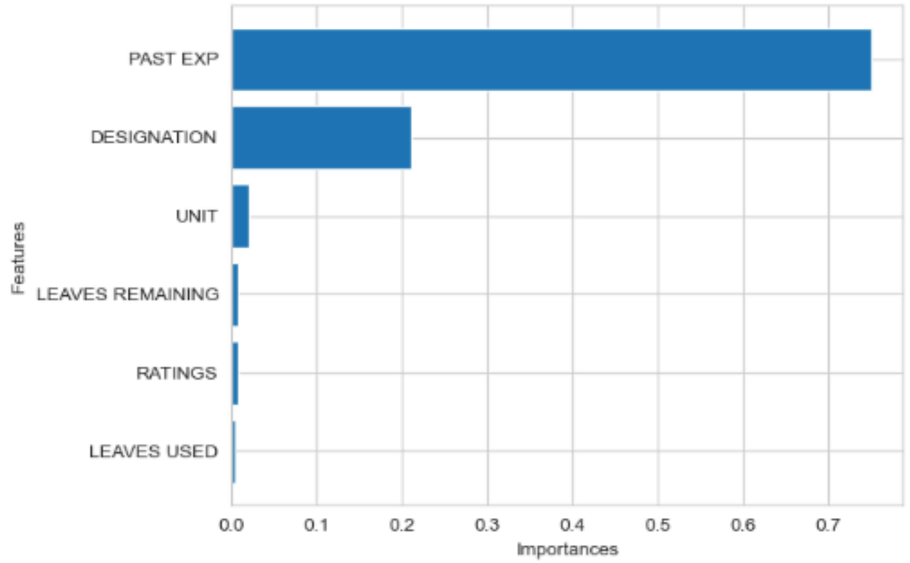
```
Mean Absolute Error:  4953.12511972531
```

Importance of Featrue on Salary (on basis of Decision Tree Regressor)

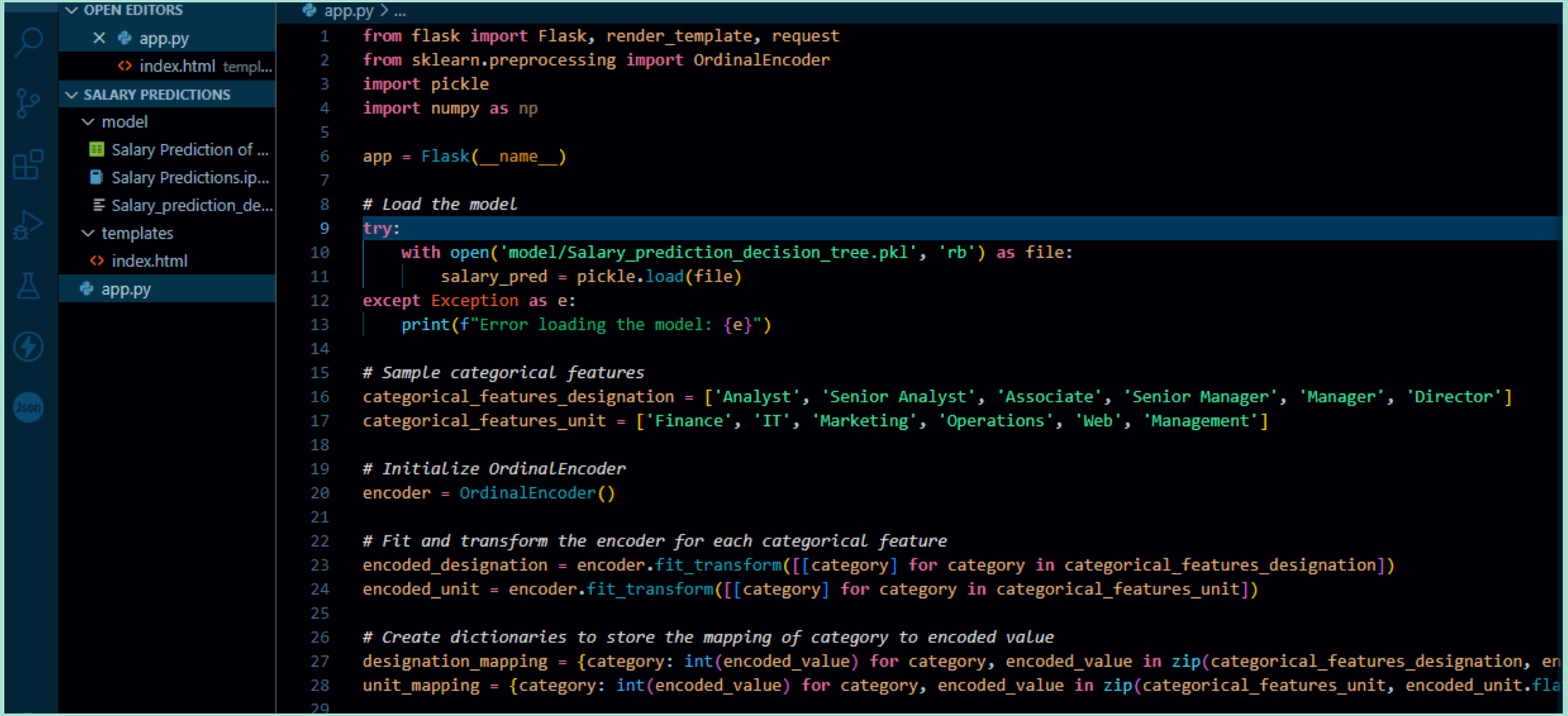
- By employing a Decision Tree Regressor for salary prediction, we can quantify the importance of each feature; for instance, "years of experience" might show a higher importance score, indicating it has a substantial impact on predicting salary, while "job title" and "education level" also contribute significantly but to a lesser extent.

Importance of Features on the Salary (on the basis of Decision Tree Regressor)

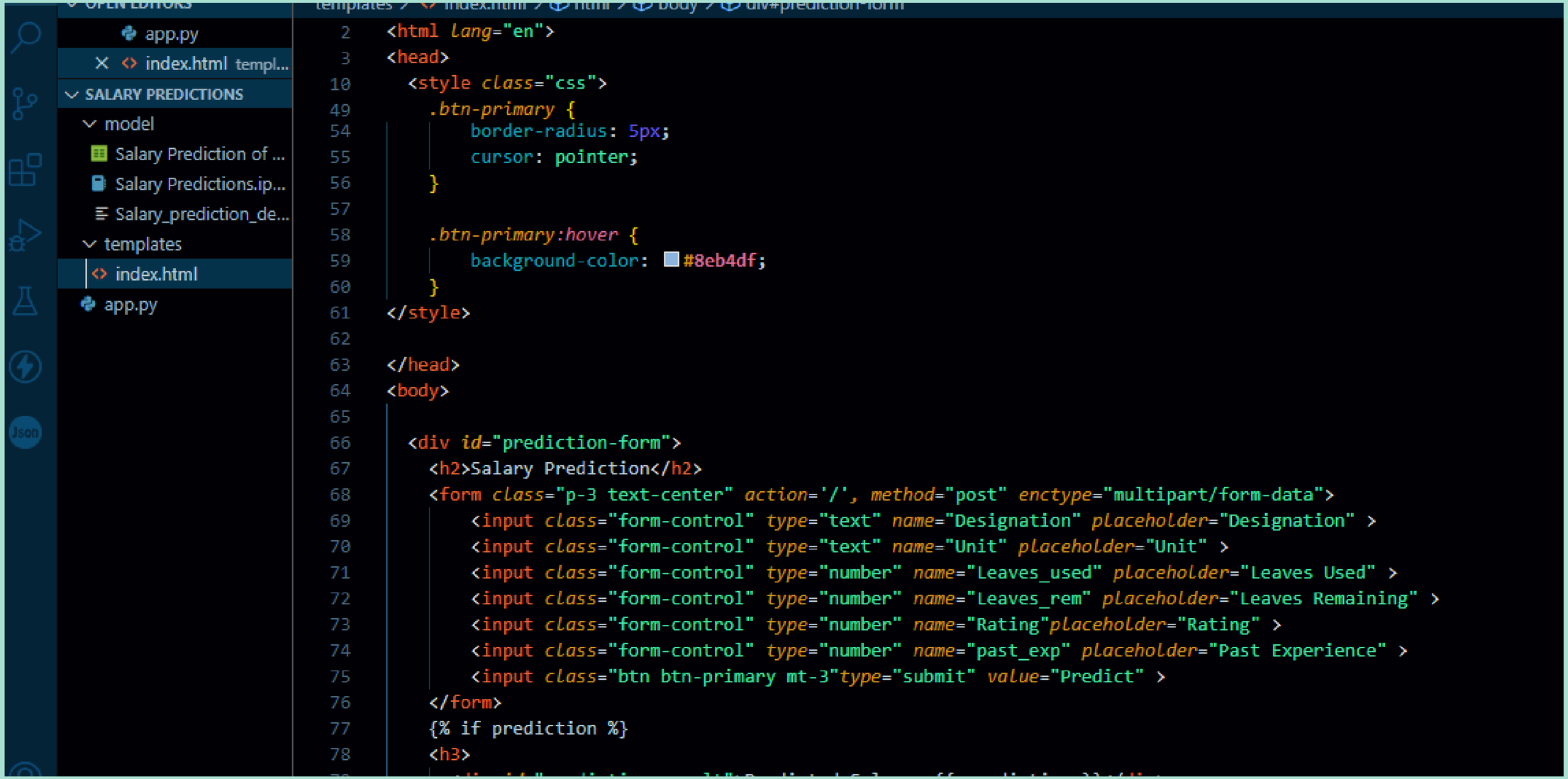
[64]:
```
feature_importance = model2.feature_importances_
feature_names = X_train.columns

feature_importance_df = pd.DataFrame({'Feature': feature_names, 'Importance': feature_importance})

feature_importance_df = feature_importance_df.sort_values(by='Importance', ascending=False)

fig,ax = plt.subplots()
ax.barh(feature_importance_df['Feature'],feature_importance_df['Importance'][:20])
ax.set_ylabel('Features')
ax.set_xlabel('Importances')
ax.invert_yaxis()
```



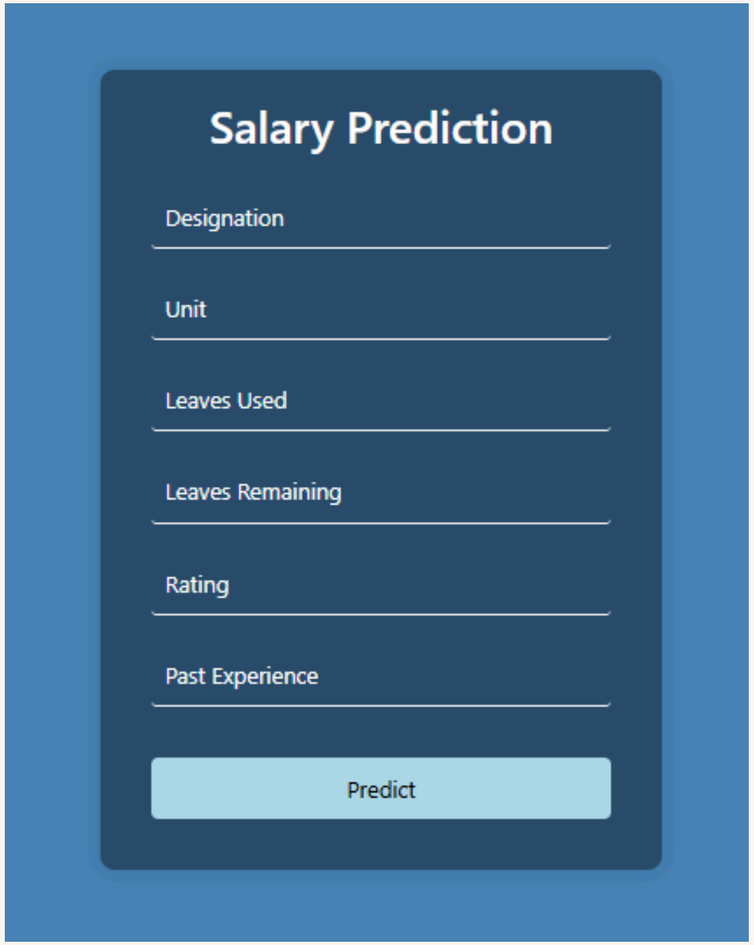
Model Deployment

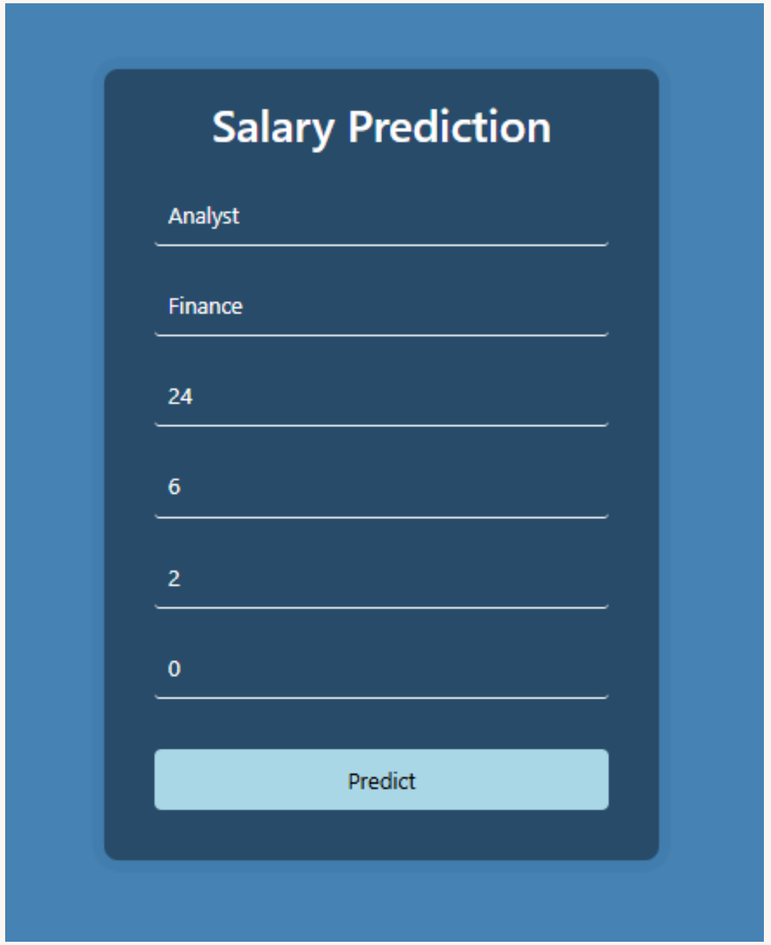


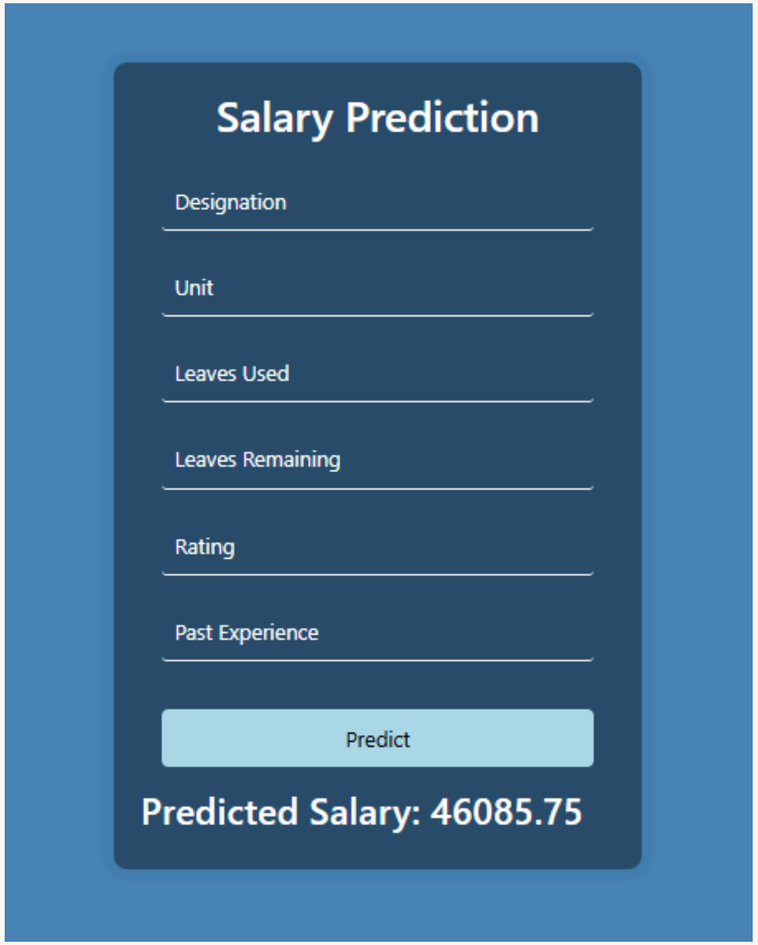
Model Deployment



ANALYSIS OUTCOME

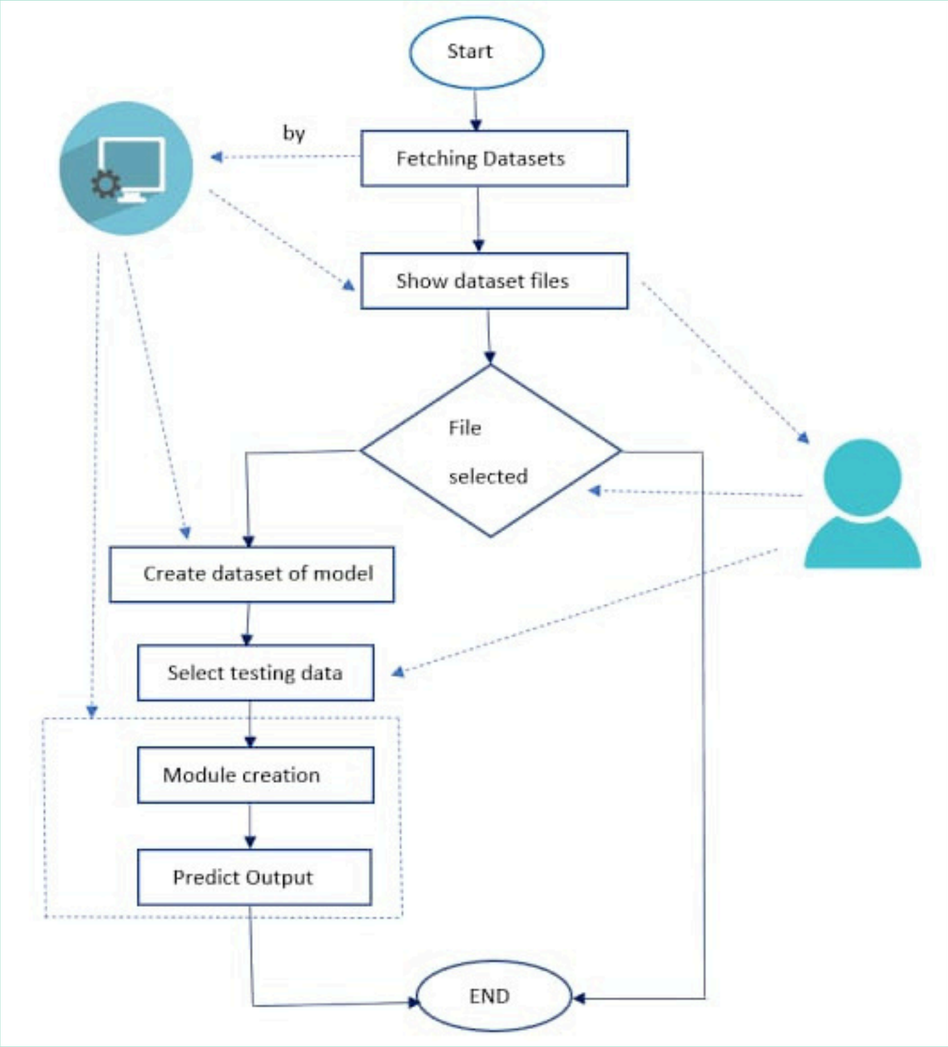






FlowChart Execution

- Start: Initiate the application.
- Receive Input Data: Collect user input (e.g., experience, education, job title).
- Validate Input Data: Check and ensure data is complete and correctly formatted. Return an error if invalid.
- Preprocess Input Data: Transform input data to the format required by the model.
- Load Trained Model: Load the pre-trained salary prediction model. Return an error if loading fails.
- Predict Salary: Use the model to predict the salary based on preprocessed data. Return an error if prediction fails.
- Return Prediction: Send the predicted salary to the user.
- End: Complete the process.



CONCLUSION

This project successfully developed a predictive model for estimating data professional salaries using machine learning. By following a structured process of data analysis, preprocessing, feature engineering, model training, and evaluation, we created an accurate and reliable prediction system. The model was deployed via a Flask application, enabling easy and efficient real-time salary predictions. This project underscores the effectiveness of data-driven solutions in enhancing decision-making in IT industry and career planning.

THANKYOU

For joining me on this journey of exploring Salary Prediction for Data Professions.
Our knowledge and skills will continue to evolve with practice and experimentation.

FOLLOW ME

https://www.linkedin.com/in/md-shakib-6283a7239/

https://github.com/shaky1405