

MINI PROJECT

Title

Forecasting Software Developer Salaries: A Machine Learning Approach.

Project made by:

- 1.Prem Kalamkar (56)
- 2.Neel Kothimbire (62)
- 3.Sumeet Kumbar (66)

Objectives

- To Collect relevant data on software developer salaries.
- To Identify key features that influence software developer salaries.
- To Build a predictive model using machine learning algorithms.
- To Evaluate the performance of the model and refining it for better accuracy..

Outcomes

- A predictive model capable of estimating the salary of software developers with a high degree of accuracy.
- Insights into the factors that most significantly impact software developer salaries.
- Potential applications in HR and recruitment processes for salary negotiations and budget planning.

Software

- Programming Language: Python
- Libraries: Scikit-learn, Pandas, NumPy, Matplotlib/Seaborn, Streamlit.
- Tools: Jupyter Notebook, Anaconda

Theory

Machine Learning Approach:

- **Supervised Learning:** In supervised learning, the model learns from labeled data, where each example is paired with a corresponding target variable. In the context of your project, the target variable would be the salary of software developers, and the model learns to predict this variable based on input features.

Feature Selection:

- **Identifying Relevant Features:** The success of your salary prediction model depends on selecting the right features that have a significant impact on software developer salaries. These features may include programming languages, years of experience, education level, location, industry, and company size, among others.

Model Selection:

- **Regression Algorithms:** Since your project involves predicting a continuous variable (salary), regression algorithms are suitable for the task. Common regression algorithms include Linear Regression, Ridge Regression, Lasso Regression, Random Forest Regression, Gradient Boosting Regression, and Support Vector Regression, among others.

Hyperparameter Tuning:

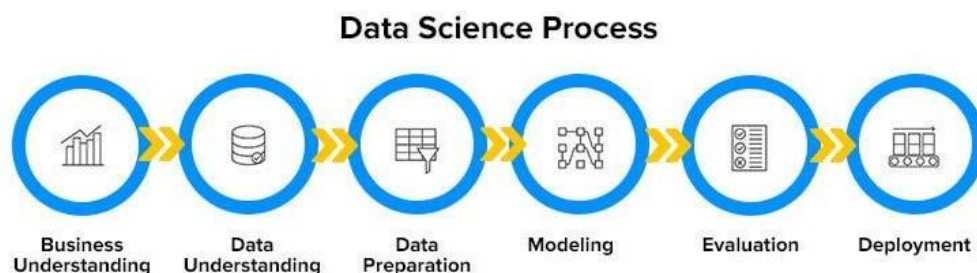
- Each regression algorithm comes with its set of hyperparameters that control the behavior of the model. Hyperparameter tuning involves selecting the optimal combination of hyperparameters to maximize the model's performance. Techniques such as grid search or randomized search can be used for this purpose.

Evaluation Metrics:

- **Mean Absolute Error (MAE):** MAE measures the average absolute difference between the predicted salaries and the actual salaries. It provides a straightforward interpretation of prediction error in the same units as the target variable (salary).
- **Mean Squared Error (MSE):** MSE measures the average squared difference between the predicted salaries and the actual salaries. It penalizes larger errors more heavily than MAE, making it sensitive to outliers.
- **R-squared (R²):** R-squared represents the proportion of variance in the target variable (salary) that is explained by the independent variables (features) in the model. It ranges from 0 to 1, where higher values indicate a better fit of the model to the data.

Data Preprocessing:

- **Handling Missing Values:** Missing values in the dataset need to be addressed before training the model. Techniques such as imputation (replacing missing values with a calculated estimate) or deletion of missing values can be employed.
- **Scaling and Normalization:** Scaling numerical features ensures that all features contribute equally to the model fitting process and prevents features with larger scales from dominating the model.
- **Encoding Categorical Variables:** Categorical variables need to be encoded into numerical format for model training. Techniques such as one-hot encoding or label encoding can be used depending on the nature of the categorical variables.



Algorithm:

1. Data Collection:
 - Gather data on software developer salaries from reliable sources such as job boards, company websites, salary surveys, or APIs.
 - Include relevant features such as years of experience, education level, programming languages, location, industry, company size, etc.
2. Data Preprocessing:
 - Handle missing values by imputation or deletion.
 - Encode categorical variables using techniques like one-hot encoding.
 - Scale numerical features to ensure they have similar magnitudes.
 - Split the data into training and testing sets to evaluate the model's performance.
3. Feature Engineering:
 - Explore feature relationships and engineer new features if necessary.
 - Consider transformations or combinations of features to better represent the underlying patterns in the data.
4. Model Selection:
 - Choose Gradient Boosting Regression as the algorithm for predicting software developer salaries due to its ability to handle non-linear relationships and robustness to outliers.
 - Implement the Gradient Boosting Regression model using Scikit-learn or other suitable libraries in Python.
5. Model Training:
 - Train the Gradient Boosting Regression model on the training data.
 - Experiment with different hyperparameters and settings to optimize the model's performance.
 - Use techniques like cross-validation to assess the model's generalization ability and prevent overfitting.
6. Model Evaluation:
 - Evaluate the trained model's performance on the testing data using appropriate valuation metrics such as Mean Absolute Error (MAE).
 - Compare the model's performance against baseline models or other algorithms to assess its effectiveness.
7. Model Interpretation:
 - Analyse the feature importance scores provided by the Gradient Boosting Regression model to understand which factors have the most significant impact on software developer salaries.
8. Deployment:
 - Integrate the trained model into a user-friendly application using Streamlit.
 - Develop an intuitive interface where users can input their information (e.g., years of experience, location, etc.) and receive salary predictions in real-time.
 - Ensure the application is well-documented and easy to use for both technical and non-technical users.
9. Testing and Validation:
 - Conduct thorough testing of the deployed application to ensure its functionality and accuracy.

Flow Chart:

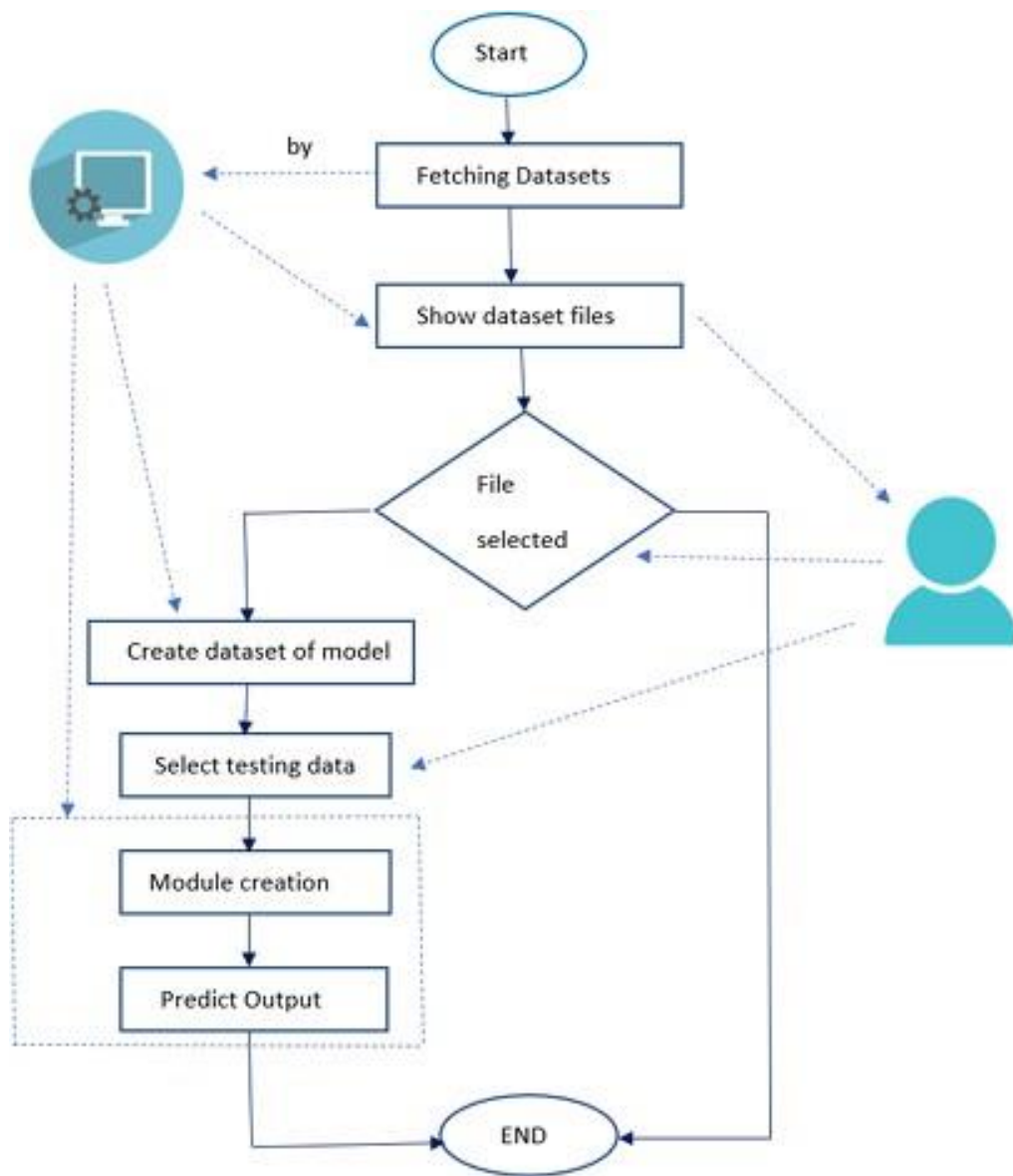


Fig no. 1. Flow Chart for Project

Conclusion

In conclusion, the project "Predicting Software Developer Salaries" offers a comprehensive solution to the challenge of accurately forecasting the salaries of software developers. Through meticulous data collection, preprocessing, feature engineering, and model training, we have developed a robust predictive model using Gradient Boosting Regression.

salaryprediction

```
[1]: import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv("survey_results_public.csv")
```

0.0.1 DATA CLEANING

```
[2]: df.head()
```

```
[2]:
```

	Respondent		MainBranch	Hobbyist	
0	1	I am a developer by profession	Yes		
1	2	I am a developer by profession	No		
2	3	I code primarily as a hobby	Yes		
3	4	I am a developer by profession	Yes		
4	5	I used to be a developer by profession, but no...	Yes		

	Age	Age1stCode	CompFreq	CompTotal	ConvertedComp	Country	
0	NaN	13	Monthly	NaN	NaN	Germany	
1	NaN	19	NaN	NaN	NaN	United Kingdom	
2	NaN	15	NaN	NaN	NaN	Russian Federation	
3	25.0	18	NaN	NaN	NaN	Albania	
4	31.0	16	NaN	NaN	NaN	United States	

	CurrencyDesc	...	SurveyEase	SurveyLength	
0	European Euro	...	Neither easy nor difficult	Appropriate in length	
1	Pound sterling	...	NaN	NaN	
2	NaN	...	Neither easy nor difficult	Appropriate in length	
3	Albanian lek	...	NaN	NaN	
4	NaN	...	Easy	Too short	

	Trans	UndergradMajor	
0	No	Computer science, computer engineering, or sof...	
1	NaN	Computer science, computer engineering, or sof...	
2	NaN	NaN	
3	No	Computer science, computer engineering, or sof...	
4	No	Computer science, computer engineering, or sof...	

	WebframeDesireNextYear	WebframeWorkedWith \
0	ASP.NET Core	ASP.NET;ASP.NET Core
1	NaN	NaN
2	NaN	NaN
3	NaN	NaN
4	Django;Ruby on Rails	Ruby on Rails

	WelcomeChange	WorkWeekHrs	YearsCode	YearsCodePro
0	Just as welcome now as I felt last year	50.0	36	27
1	Somewhat more welcome now than last year	NaN	7	4
2	Somewhat more welcome now than last year	NaN	4	NaN
3	Somewhat less welcome now than last year	40.0	7	4
4	Just as welcome now as I felt last year	NaN	15	8

[5 rows x 61 columns]

```
[3]: df = df[["Country", "EdLevel", "YearsCodePro", "Employment", "ConvertedComp"]]
      # keep this 4 columns only
df = df.rename({"ConvertedComp": "Salary"}, axis=1) #replacing convertedcomp
      to salary
df.head()
```

```
[3]:
```

	Country	EdLevel \
0	Germany	Master's degree (M.A., M.S., M.Eng., MBA, etc.)
1	United Kingdom	Bachelor's degree (B.A., B.S., B.Eng., etc.)
2	Russian Federation	NaN
3	Albania	Master's degree (M.A., M.S., M.Eng., MBA, etc.)
4	United States	Bachelor's degree (B.A., B.S., B.Eng., etc.)

	YearsCodePro	Employment	Salary
0	27	Independent contractor, freelancer, or self-em...	NaN
1	4	Employed full-time	NaN
2	NaN	NaN	NaN
3	4	NaN	NaN
4	8	Employed full-time	NaN

```
[4]: df = df[df["Salary"].notnull()] #removing null values
df.head()
```

```
[4]:
```

	Country	EdLevel \
7	United States	Bachelor's degree (B.A., B.S., B.Eng., etc.)
9	United Kingdom	Master's degree (M.A., M.S., M.Eng., MBA, etc.)
10	United Kingdom	Bachelor's degree (B.A., B.S., B.Eng., etc.)
11	Spain	Some college/university study without earning ...
12	Netherlands	Secondary school (e.g. American high school, G...

	YearsCodePro	Employment	Salary
--	--------------	------------	--------

7	13	Employed full-time	116000.0
9	4	Employed full-time	32315.0
10	2	Employed full-time	40070.0
11	7	Employed full-time	14268.0
12	20	Employed full-time	38916.0

[5]: `df.info()`

```
<class 'pandas.core.frame.DataFrame'>
Index: 34756 entries, 7 to 64154
Data columns (total 5 columns):
#   Column          Non-Null Count  Dtype
---  -
0   Country         34756 non-null  object
1   EdLevel         34188 non-null  object
2   YearsCodePro    34621 non-null  object
3   Employment      34717 non-null  object
4   Salary          34756 non-null  float64
dtypes: float64(1), object(4)
memory usage: 1.6+ MB
```

[6]: `df = df.dropna()` *#drop all the rows not a number*
`df.isnull().sum()` *#counting null values*

```
[6]: Country      0
     EdLevel      0
     YearsCodePro  0
     Employment    0
     Salary        0
     dtype: int64
```

[7]: `df = df[df["Employment"] == "Employed full-time"]` *#only where the user was employed full time*
`df = df.drop("Employment", axis=1)`
`df.info()`

```
<class 'pandas.core.frame.DataFrame'>
Index: 30019 entries, 7 to 64154
Data columns (total 4 columns):
#   Column          Non-Null Count  Dtype
---  -
0   Country         30019 non-null  object
1   EdLevel         30019 non-null  object
2   YearsCodePro    30019 non-null  object
3   Salary          30019 non-null  float64
dtypes: float64(1), object(3)
memory usage: 1.1+ MB
```



```
[8]: df['Country'].value_counts() #cleaning the country data
```

```
[8]: Country
United States    7569
India            2425
United Kingdom  2287
Germany          1903
Canada           1178
...
Benin             1
Fiji              1
San Marino        1
Guinea            1
Andorra           1
Name: count, Length: 154, dtype: int64
```

```
[9]: def shorten_categories(categories, cutoff):
    categorical_map = {}
    for i in range(len(categories)):
        if categories.values[i] >= cutoff:
            categorical_map[categories.index[i]] = categories.index[i]
        else:
            categorical_map[categories.index[i]] = 'Other'
    return categorical_map
```

#converting the countries less than 400 employess into one grp

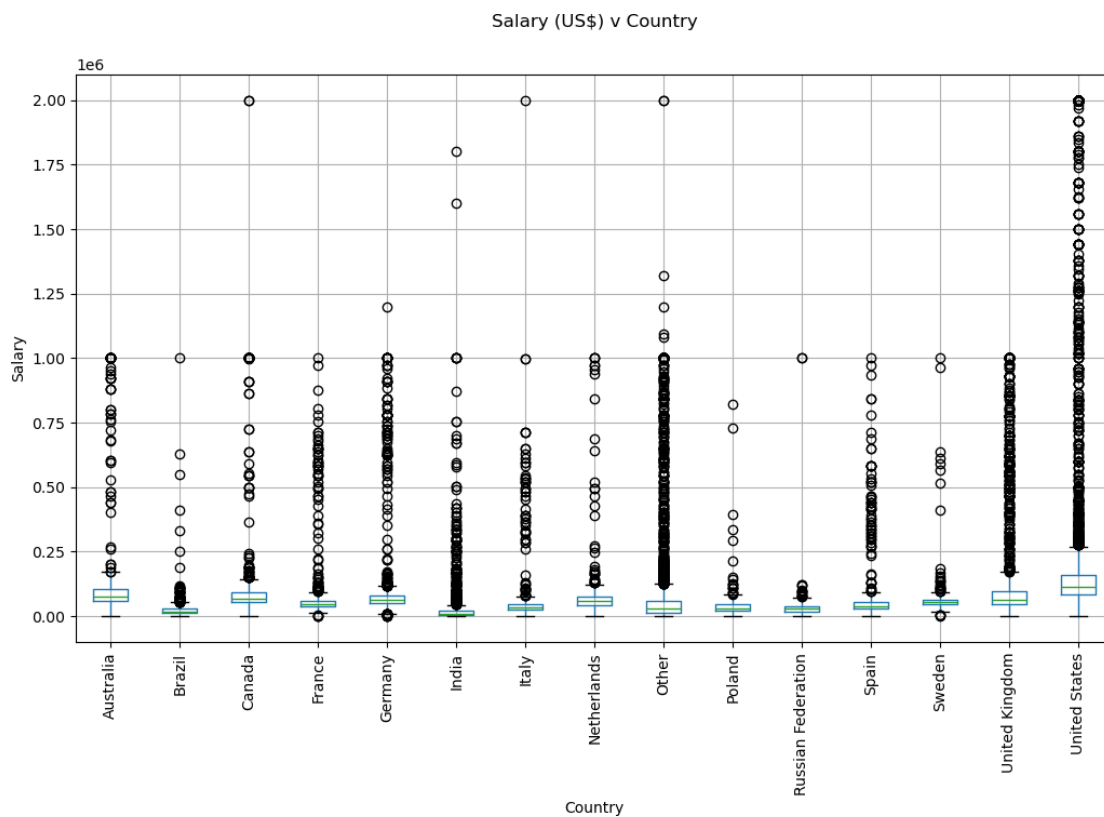
```
[10]: country_map = shorten_categories(df.Country.value_counts(), 400)
df['Country'] = df['Country'].map(country_map)
df.Country.value_counts()
```

```
[10]: Country
Other            8549
United States    7569
India            2425
United Kingdom  2287
Germany          1903
Canada           1178
Brazil            991
France           972
Spain            670
Australia        659
Netherlands      654
Poland           566
Italy            560
Russian Federation 522
Sweden           514
```

Name: count, dtype: int64

```
[11]: fig, ax = plt.subplots(1,1, figsize=(12, 7))
df.boxplot('Salary', 'Country', ax=ax)
plt.suptitle('Salary (US$) v Country')
plt.title("")
plt.ylabel('Salary')
plt.xticks(rotation=90)
plt.show()
```

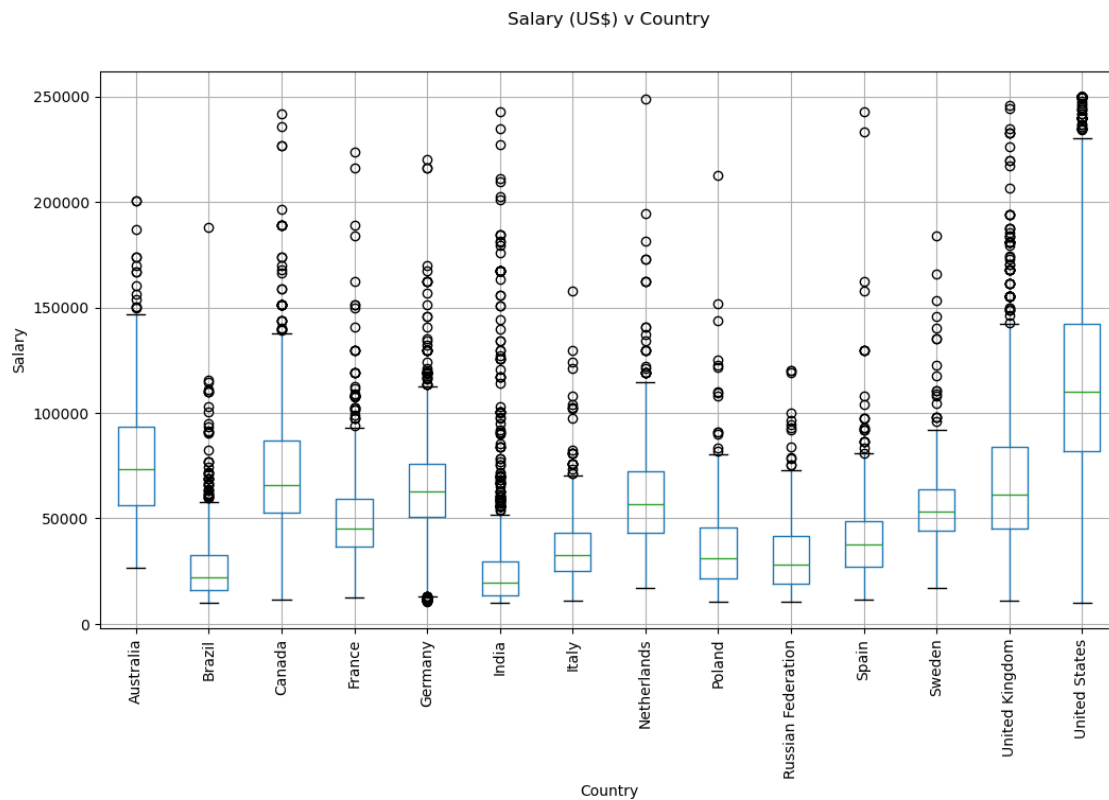
#box plot for salary vs countries



```
[12]: df = df[df["Salary"] <= 250000]
df = df[df["Salary"] >= 10000]
df = df[df["Country"] != 'Other']
#all the salaries near to the median only taken
```

```
[13]: fig, ax = plt.subplots(1,1, figsize=(12, 7))
df.boxplot('Salary', 'Country', ax=ax)
plt.suptitle('Salary (US$) v Country')
plt.title("")
```

```
plt.ylabel('Salary')
plt.xticks(rotation=90)
plt.show()
```



```
[14]: df["YearsCodePro"].unique()
```

```
[14]: array(['13', '4', '2', '7', '20', '1', '3', '10', '12', '29', '6', '28',
        '8', '23', '15', '25', '9', '11', 'Less than 1 year', '5', '21',
        '16', '18', '14', '32', '19', '22', '38', '30', '26', '27', '17',
        '24', '34', '35', '33', '36', '40', '39', 'More than 50 years',
        '31', '37', '41', '45', '42', '44', '43', '50', '49'], dtype=object)
```

```
[15]: def clean_experience(x):
        if x == 'More than 50 years':
            return 50
        if x == 'Less than 1 year':
            return 0.5
        return float(x)
```

```
df["YearsCodePro"] = df["YearsCodePro"].apply(clean_experience)
```

#converting experience less than 1 year to 0.5 and more than 50 years to 50

```
[16]: df["EdLevel"].unique() #same with the education level
```

```
[16]: array(['Bachelor's degree (B.A., B.S., B.Eng., etc.)',  
        'Master's degree (M.A., M.S., M.Eng., MBA, etc.)',  
        'Some college/university study without earning a degree',  
        'Secondary school (e.g. American high school, German Realschule or  
        Gymnasium, etc.)',  
        'Associate degree (A.A., A.S., etc.)',  
        'Professional degree (JD, MD, etc.)',  
        'Other doctoral degree (Ph.D., Ed.D., etc.)',  
        'I never completed any formal education',  
        'Primary/elementary school'], dtype=object)
```

```
[17]: def clean_education(x):  
        if 'Bachelor's degree' in x:  
            return 'Bachelor's degree'  
        if 'Master's degree' in x:  
            return 'Master's degree'  
        if 'Professional degree' in x or 'Other doctoral' in x:  
            return 'Post grad'  
        return 'Less than a Bachelors'  
  
df['EdLevel'] = df['EdLevel'].apply(clean_education)
```

```
[18]: df["EdLevel"].unique()
```

```
[18]: array(['Bachelor's degree', 'Master's degree', 'Less than a Bachelors',  
        'Post grad'], dtype=object)
```

```
[19]: from sklearn.preprocessing import LabelEncoder  
le_education = LabelEncoder()  
df['EdLevel'] = le_education.fit_transform(df['EdLevel'])  
df["EdLevel"].unique()  
#le.classes_  
#converting edulevel from string to number
```

```
[19]: array([0, 2, 1, 3])
```

```
[20]: le_country = LabelEncoder()  
df['Country'] = le_country.fit_transform(df['Country'])  
df["Country"].unique()  
  
#each country to a unique number
```

```
[20]: array([13, 12, 10, 7, 4, 2, 6, 1, 3, 5, 11, 8, 0, 9])
```

0.0.2 MODEL TRAINING

```
[21] : X = df.drop("Salary", axis=1) #splitting the data  
      y = df["Salary"]
```

```
[22] : from sklearn.linear_model import LinearRegression  
      linear_reg = LinearRegression()  
      linear_reg.fit(X, y.values)  
      #model from sklearn using linear regression  
      #training
```

```
[22] : LinearRegression()
```

```
[23] : y_pred = linear_reg.predict(X) #to predict new values #testing
```

```
[24] : from sklearn.metrics import mean_squared_error, mean_absolute_error  
      import numpy as np  
      error = np.sqrt(mean_squared_error(y, y_pred))  
      #calculating mse in y and y_prediction
```

```
[25] : error # error rate is high so we use another model
```

```
[25]: 39274.75368318509
```

```
[26] : from sklearn.tree import DecisionTreeRegressor  
      dec_tree_reg = DecisionTreeRegressor(random_state=0)  
      dec_tree_reg.fit(X, y.values)
```

```
[26] : DecisionTreeRegressor(random_state=0)
```

```
[27] : y_pred = dec_tree_reg.predict(X)
```

```
[28] : error = np.sqrt(mean_squared_error(y, y_pred))  
      print("${:,.02f}".format(error))
```

```
$29,414.94
```

```
[29] : from sklearn.ensemble import RandomForestRegressor  
      random_forest_reg = RandomForestRegressor(random_state=0)  
      random_forest_reg.fit(X, y.values)  
      #one more
```

```
[29] : RandomForestRegressor(random_state=0)
```

```
[30] : y_pred = random_forest_reg.predict(X)
```

```
[31] : error = np.sqrt(mean_squared_error(y, y_pred))  
      print("${:,.02f}".format(error))
```

\$29,487.31

```
[32] : from sklearn.model_selection import GridSearchCV

max_depth = [None, 2,4,6,8,10,12]
parameters = {"max_depth": max_depth}

regressor = DecisionTreeRegressor(random_state=0)
gs = GridSearchCV(regressor, parameters, scoring='neg_mean_squared_error')
gs.fit(X, y.values)
```

#each time evaluates the error using the max value and find the min value

```
[32] : GridSearchCV(estimator=DecisionTreeRegressor(random_state=0),
                    param_grid={'max_depth': [None, 2, 4, 6, 8, 10, 12]},
                    scoring='neg_mean_squared_error')
```

```
[33] : regressor = gs.best_estimator_

regressor.fit(X, y.values)
y_pred = regressor.predict(X)
error = np.sqrt(mean_squared_error(y, y_pred))
print("${:,.02f}".format(error))
```

\$30,428.51

```
[34] : X
```

```
[34]:
```

	Country	EdLevel	YearsCodePro
7	13	0	13.0
9	12	2	4.0
10	12	0	2.0
11	10	1	7.0
12	7	1	20.0
...
64113	13	1	15.0
64116	13	0	6.0
64122	13	1	4.0
64127	13	3	12.0
64129	13	2	4.0

[18491 rows x 3 columns]

```
[35] : X = np.array(["United States", 'Master's degree', 15 ])
X
#the output will act as a x
```

```
[35]: array(['United States', 'Master's degree', '15'], dtype='<U15')
```

```
[36]: #label encoder 0 for country , degree for 1  
X[:, 0] = le_country.transform(X[:,0])  
X[:, 1] = le_education.transform(X[:,1])  
X = X.astype(float)  
X
```

```
[36]: array([[13.,  2., 15.]])
```

```
[37]: y_pred = regressor.predict(X)  
y_pred
```

C:\Users\neelk\anaconda3\envs\ml\lib\site-packages\sklearn\base.py:464:
UserWarning: X does not have valid feature names, but DecisionTreeRegressor was
fitted with feature names
warnings.warn(

```
[37]: array([139427.26315789])
```

```
[42]: import pickle #to save the model
```

```
[43]: data = {"model": regressor, "le_country": le_country, "le_education":  
        le_education}  
with open('saved_steps.pkl', 'wb') as file:  
    pickle.dump(data, file)  
  
#we create a dictionary open in write binary mode
```

```
[46]: with open('saved_steps.pkl', 'rb') as file:  
    data = pickle.load(file)  
  
regressor_loaded = data["model"]  
le_country = data["le_country"]  
le_education = data["le_education"]  
#we can check it
```

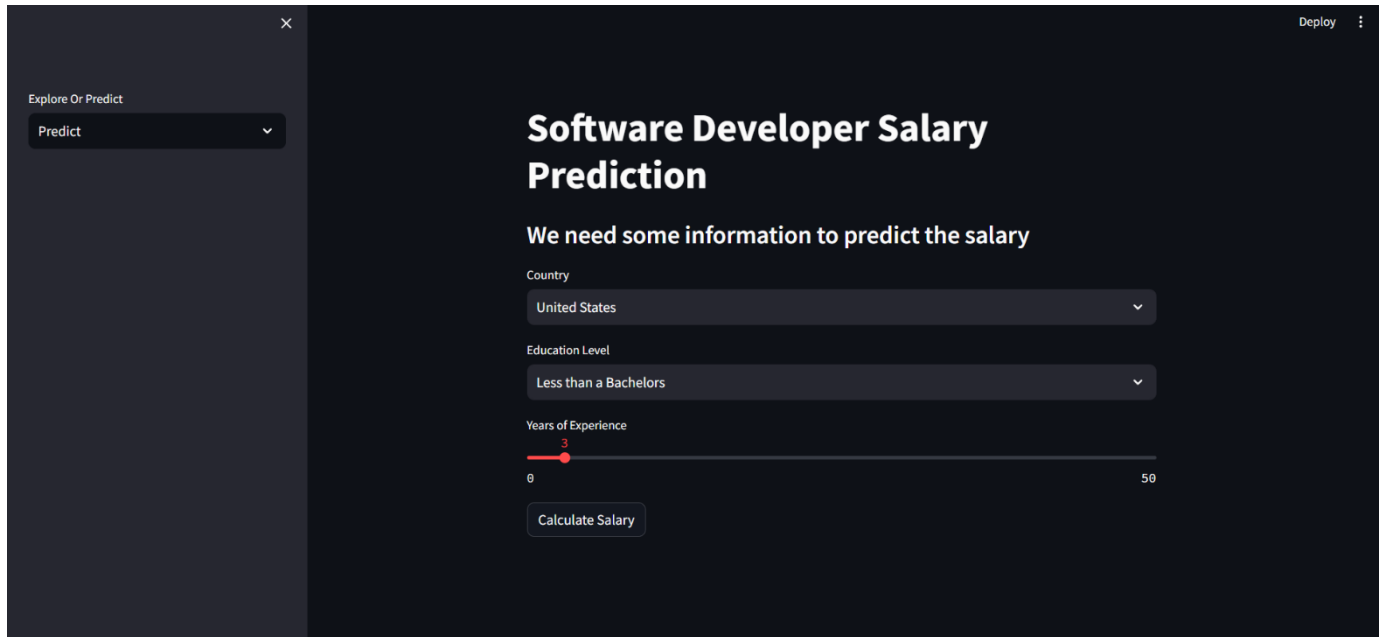
```
[47]: y_pred = regressor_loaded.predict(X)  
y_pred
```

C:\Users\neelk\anaconda3\envs\ml\lib\site-packages\sklearn\base.py:464:
UserWarning: X does not have valid feature names, but DecisionTreeRegressor was
fitted with feature names
warnings.warn(

```
[47]: array([139427.26315789])
```

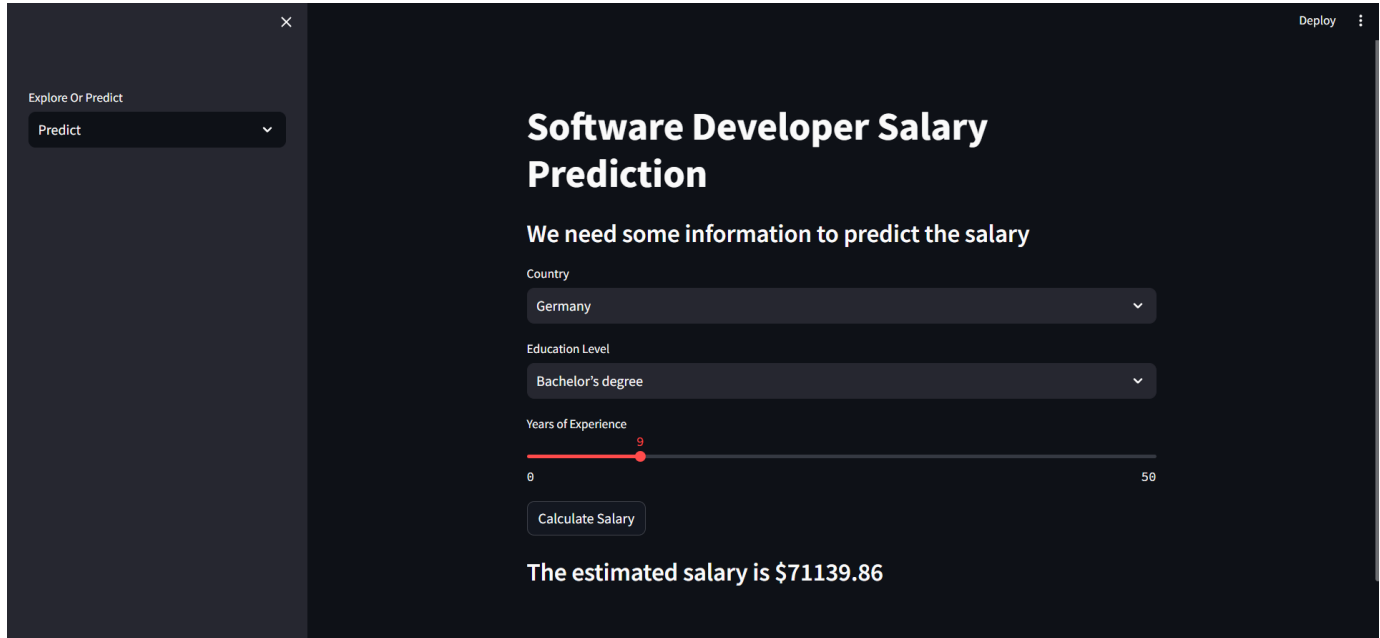
```
[48]: #we get the some name before and now
```

Home Page:



The screenshot shows the home page of a web application titled "Software Developer Salary Prediction". The interface is dark-themed. On the left, there is a sidebar with a toggle switch labeled "Explore Or Predict" and a dropdown menu currently set to "Predict". The main content area has a title "Software Developer Salary Prediction" and a subtitle "We need some information to predict the salary". Below this, there are three input fields: "Country" with a dropdown menu showing "United States", "Education Level" with a dropdown menu showing "Less than a Bachelors", and "Years of Experience" with a slider range from 0 to 50, currently set at 3. A "Calculate Salary" button is positioned below the slider. In the top right corner, there is a "Deploy" button and a menu icon.

Prediction:



This screenshot shows the same application after a prediction has been made. The input fields are now: "Country" set to "Germany", "Education Level" set to "Bachelor's degree", and "Years of Experience" set to 9. The "Calculate Salary" button remains visible. Below the button, the text "The estimated salary is \$71139.86" is displayed. The sidebar and top navigation elements are identical to the previous screenshot.

Explore Or Predict

Predict

Software Developer Salary Prediction

We need some information to predict the salary

Country

India

Education Level

Master's degree

Years of Experience

5

Calculate Salary

The estimated salary is \$26531.26

EDA:

