Salary prediction with supervised learning Project

Madhumita Mondal

TASK:

This project involves analyzing a survey dataset of developers. Key steps include data cleaning, filtering, and visualization. A Linear Regression model, Decision Tree Regression model, and Random Forest Regression model are trained to predict salaries based on features like country, education level, and years of professional coding experience. The best-performing model is saved and can be loaded for making salary predictions on new data.

The project involves a machine learning task focused on predicting salaries for developers based on certain features. The primary goal is to analyze a dataset containing information about developers, including their country, education level, and years of professional coding experience. The type of learning employed includes supervised learning, specifically regression, where the algorithm learns to predict a continuous target variable (in this case, salaries) based on input features. The project utilizes algorithms such as Linear Regression, Decision Tree Regression, and Random Forest Regression to achieve the task of predicting developer salaries. The overarching aim is to understand the factors influencing salary variations among developers and to create models that can accurately estimate salaries for new data points.

GOAL:

The goal of the project is to analyze a dataset of developers and build machine learning models to predict salaries based on various factors such as country, education level, and years of professional coding experience. The project aims to explore patterns in developer salaries, understand the impact of different features, and create predictive models that can be used to estimate salaries for new data. The underlying objectives may include understanding the factors influencing developer salaries and creating a tool for predicting salaries in the context of the given dataset. The project could be valuable for gaining insights into the factors contributing to salary variations among developers and for developing a practical application of machine learning in the domain of compensation prediction for software developers.

Data:

The dataset comprises survey responses from 64,461 individuals, with each respondent contributing information to 61 columns. Data collected from Kaggle.

Below are some key details about the data:

Data Size:

• 64,461 entries, 61 columns.

Types of Data:

• Numerical Data Types: int64 (1 column), float64 (4 columns).

Categorical/Object Data Types:

object (56 columns), representing various categorical features such as employment status, programming languages used, education level, etc.

Missing Values:

• The dataset contains missing values in multiple columns, with varying degrees of completeness.

Features of Interest:

• Features include respondent details like age, country, compensation details, job satisfaction, and information about technologies used and desired.

Data Types:

• The YearsCode and YearsCodePro columns, which likely represent the number of years a respondent has been coding overall and professionally, are currently of type object and may need conversion to numeric types for analysis.

It's important to note that handling missing values, converting appropriate columns to the correct data types, and exploring the distribution of features are common steps in the preprocessing phase before analysis or model training. Additionally, understanding the meaning and context of each column is crucial for accurate interpretation and modeling.

In [995…

```
# Import necessary libraries
import pandas as pd
import matplotlib.pyplot as plt

# Load the dataset from a CSV file
df = pd.read_csv("survey_results_public.csv")
```

In [996…

```
df.head()
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 64461 entries, 0 to 64460
Data columns (total 61 columns):
 #   Column                        Non-Null Count  Dtype  
---  ------                        --------------  -----  
 0   Respondent                    64461 non-null  int64  
 1   MainBranch                    64162 non-null  object 
 2   Hobbyist                      64416 non-null  object 
 3   Age                           45446 non-null  float64
 4   Age1stCode                    57900 non-null  object 
 5   CompFreq                      40069 non-null  object 
 6   CompTotal                     34826 non-null  float64
 7   ConvertedComp                 34756 non-null  float64
 8   Country                       64072 non-null  object 
 9   CurrencyDesc                  45472 non-null  object 
 10  CurrencySymbol                45472 non-null  object 
 11  DatabaseDesireNextYear        44070 non-null  object 
 12  DatabaseWorkedWith            49537 non-null  object 
 13  DevType                       49370 non-null  object 
 14  EdLevel                       57431 non-null  object 
 15  Employment                    63854 non-null  object 
 16  Ethnicity                     45948 non-null  object 
 17  Gender                        50557 non-null  object 
 18  JobFactors                    49349 non-null  object 
 19  JobSat                        45194 non-null  object 
 20  JobSeek                       51727 non-null  object 
 21  LanguageDesireNextYear        54113 non-null  object 
 22  LanguageWorkedWith            57378 non-null  object 
 23  MiscTechDesireNextYear        42379 non-null  object 
 24  MiscTechWorkedWith            40314 non-null  object 
 25  NEWCollabToolsDesireNextYear  47287 non-null  object 
 26  NEWCollabToolsWorkedWith      52883 non-null  object 
 27  NEWDevOps                     42686 non-null  object 
 28  NEWDevOpsImpt                 41732 non-null  object 
 29  NEWEdImpt                     48465 non-null  object 
 30  NEWJobHunt                    42286 non-null  object 
 31  NEWJobHuntResearch            41022 non-null  object 
 32  NEWLearn                      56156 non-null  object 
 33  NEWOffTopic                   50804 non-null  object 
 34  NEWOnboardGood                42623 non-null  object 
 35  NEWOtherComms                 57205 non-null  object 
 36  NEWOvertime                   43231 non-null  object 
 37  NEWPurchaseResearch           37321 non-null  object 
 38  NEWPurpleLink                 54803 non-null  object 
 39  NEWSOSites                    58275 non-null  object 
 40  NEWStuck                      54983 non-null  object 
 41  OpSys                         56228 non-null  object 
 42  OrgSize                       44334 non-null  object 
 43  PlatformDesireNextYear        50605 non-null  object 
 44  PlatformWorkedWith            53843 non-null  object 
 45  PurchaseWhat                  39364 non-null  object 
 46  Sexuality                     43992 non-null  object 
 47  SOAccount                     56805 non-null  object 
 48  SOComm                        56476 non-null  object 
 49  SOPartFreq                    46792 non-null  object 
 50  SOVisitFreq                   56970 non-null  object 
 51  SurveyEase                    51802 non-null  object 
 52  SurveyLength                  51701 non-null  object 
 53  Trans                         49345 non-null  object 
 54  UndergradMajor                50995 non-null  object 
```

```
 55  WebframeDesireNextYear        40024 non-null  object
 56  WebframeWorkedWith            42279 non-null  object
 57  WelcomeChange                 52683 non-null  object
 58  WorkWeekHrs                   41151 non-null  float64
 59  YearsCode                     57684 non-null  object
 60  YearsCodePro                  46349 non-null  object
dtypes: float64(4), int64(1), object(56)
memory usage: 30.0+ MB
```

In [997…

```python
# Select specific columns of interest
df = df[["Country", "EdLevel", "YearsCodePro", "Employment", "ConvertedComp"]]
# Rename the "ConvertedComp" column to "Salary"
df = df.rename({"ConvertedComp": "Salary"}, axis=1)
df.head()
```

Out[997]:

	Country	EdLevel	YearsCodePro	Employment	Salary
0	Germany	Master's degree (M.A., M.S., M.Eng., MBA, etc.)	27	Independent contractor, freelancer, or self-em...	NaN
1	United Kingdom	Bachelor's degree (B.A., B.S., B.Eng., etc.)	4	Employed full-time	NaN
2	Russian Federation	NaN	NaN	NaN	NaN
3	Albania	Master's degree (M.A., M.S., M.Eng., MBA, etc.)	4	NaN	NaN
4	United States	Bachelor's degree (B.A., B.S., B.Eng., etc.)	8	Employed full-time	NaN

In [998…

```python
# Remove rows with missing salary values
df = df[df["Salary"].notnull()]
df.head()
```

Out[998]:

	Country	EdLevel	YearsCodePro	Employment	Salary
7	United States	Bachelor's degree (B.A., B.S., B.Eng., etc.)	13	Employed full-time	116000.0
9	United Kingdom	Master's degree (M.A., M.S., M.Eng., MBA, etc.)	4	Employed full-time	32315.0
10	United Kingdom	Bachelor's degree (B.A., B.S., B.Eng., etc.)	2	Employed full-time	40070.0
11	Spain	Some college/university study without earning ...	7	Employed full-time	14268.0
12	Netherlands	Secondary school (e.g. American high school, G...	20	Employed full-time	38916.0

In [999…

```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 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
```

In [100…

```
df = df.dropna()
df.isnull().sum()
```

Out[1000]:

```
Country         0
EdLevel         0
YearsCodePro    0
Employment      0
Salary          0
dtype: int64
```

In [100…

```
# Keep only rows where respondents are employed full-time
df = df[df["Employment"] == "Employed full-time"]
# Remove the "Employment" column as it's no longer needed
df = df.drop("Employment", axis=1)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 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
```

In [100…

```
df['Country'].value_counts()
```

Out[1002]:

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

Data Cleaning Summary:

Initial Exploration:

• Loaded the dataset into a DataFrame using pandas. • Displayed the first few rows to get an overview of the data.

Handling Missing Values:

• Checked for missing values using df.info(). • Identified columns with missing values, such as "Age," "Age1stCode," "CompFreq," etc. • Decided to drop rows where the target variable ("ConvertedComp") has missing values, as predicting salary is the primary goal, and rows without salary information would not contribute to this task.

Column Selection:

• Initially selected specific columns of interest for analysis, including "Country," "EdLevel," "YearsCodePro," "Employment," and "ConvertedComp." • Renamed the "ConvertedComp" column to "Salary" for clarity.

Filtering Employment Status:

• Kept only rows where respondents are employed full-time, as the analysis focuses on this employment category. • Dropped the "Employment" column as it was no longer needed.

Country Data Transformation:

• Grouped less frequent countries into an "Other" category to simplify analysis.

Further Data Filtering:

• Removed outliers by restricting salary values to a range between $10,000 and$250,000.

Data Transformation - Years of Professional Coding:

• Cleaned and transformed the "YearsCodePro" column to numerical values.

Data Transformation - Education Level:

• Cleaned and transformed the "EdLevel" column into categorical values.

Label Encoding:

• Applied label encoding to categorical features using scikit-learn's LabelEncoder.

In [100…

```
# Clean and transform the "Country" column by grouping less frequent countries into "C
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
```

In [100…

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

Out[1004]:

```
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: Country, dtype: int64
```

Visualizations:

• Created a boxplot to visualize the distribution of salaries across different countries.

• Plot the correlation matrix of Dataframe using seaborn

• Plot the Histogram of Salary

• Feature Importance Plot for Random Forest Regression

A boxplot of salary vs. country can provide insights into the distribution of salaries across different countries. Here's how you can create and interpret the boxplot:

Explanation:

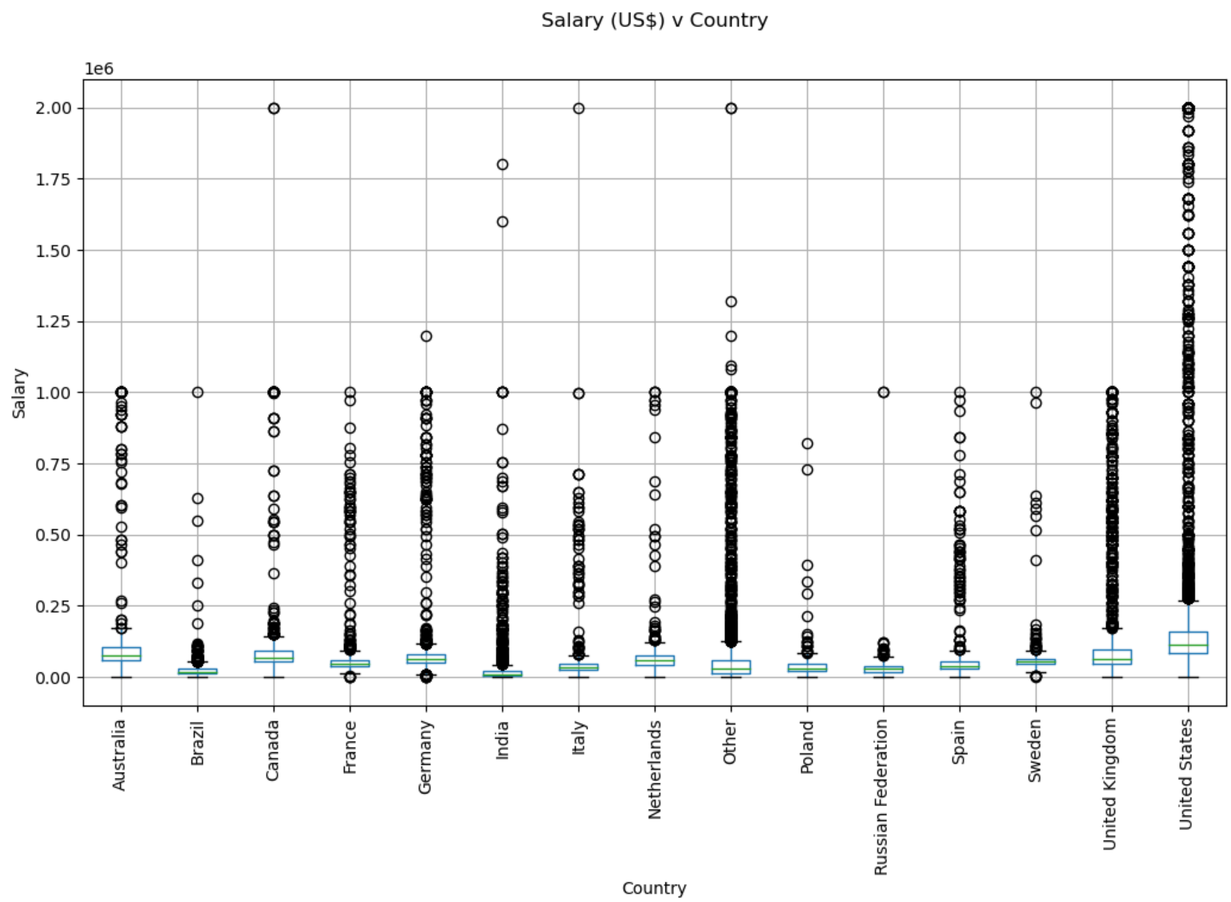
The x-axis represents the countries, and the y-axis represents the corresponding salaries. Each box in the plot represents the interquartile range (IQR) of salaries for a specific country. The line inside the box represents the median salary for each country. Whiskers extend to show the range of salaries within 1.5 times the IQR, and points beyond the whiskers are considered outliers.

Interpretation:

By examining the boxplot, you can observe the central tendency, spread, and potential outliers in salary distributions across different countries. A wider box or longer whiskers may indicate greater variability in salaries for that country. Median lines allow for a quick comparison of the central tendency of salaries in each country. Outliers may be visible as individual points beyond the whiskers, indicating unusually high or low salaries. This visualization helps in understanding the variation in salaries among different countries and identifying potential factors influencing salary differences.

In [100…

```
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()
```

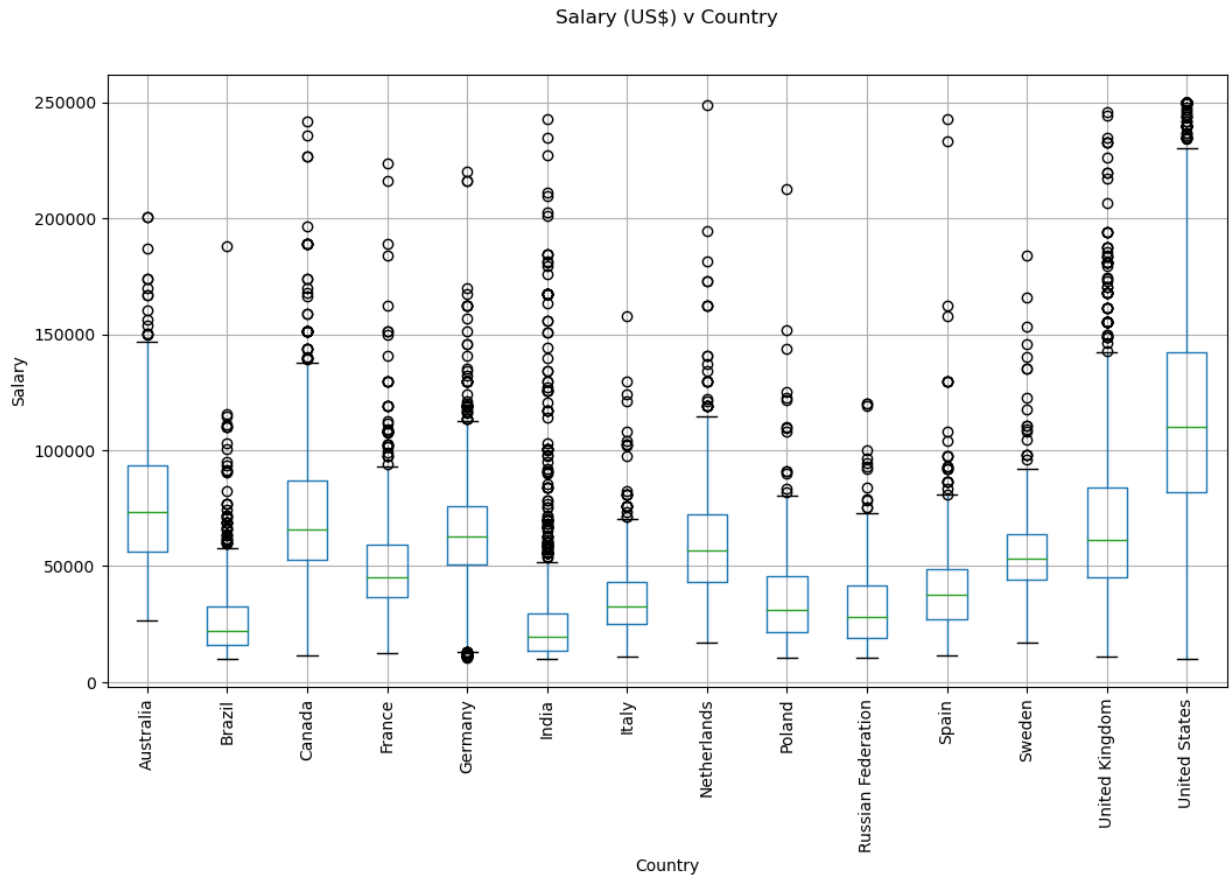


In [100…

```
df = df[df["Salary"] <= 250000]
df = df[df["Salary"] >= 10000]
df = df[df['Country'] != 'Other']
```

In [100…

```
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()
```



Correlation Matrix:

Purpose: The correlation matrix visually represents the relationships between numerical variables, helping to identify patterns and dependencies.

Explanation:

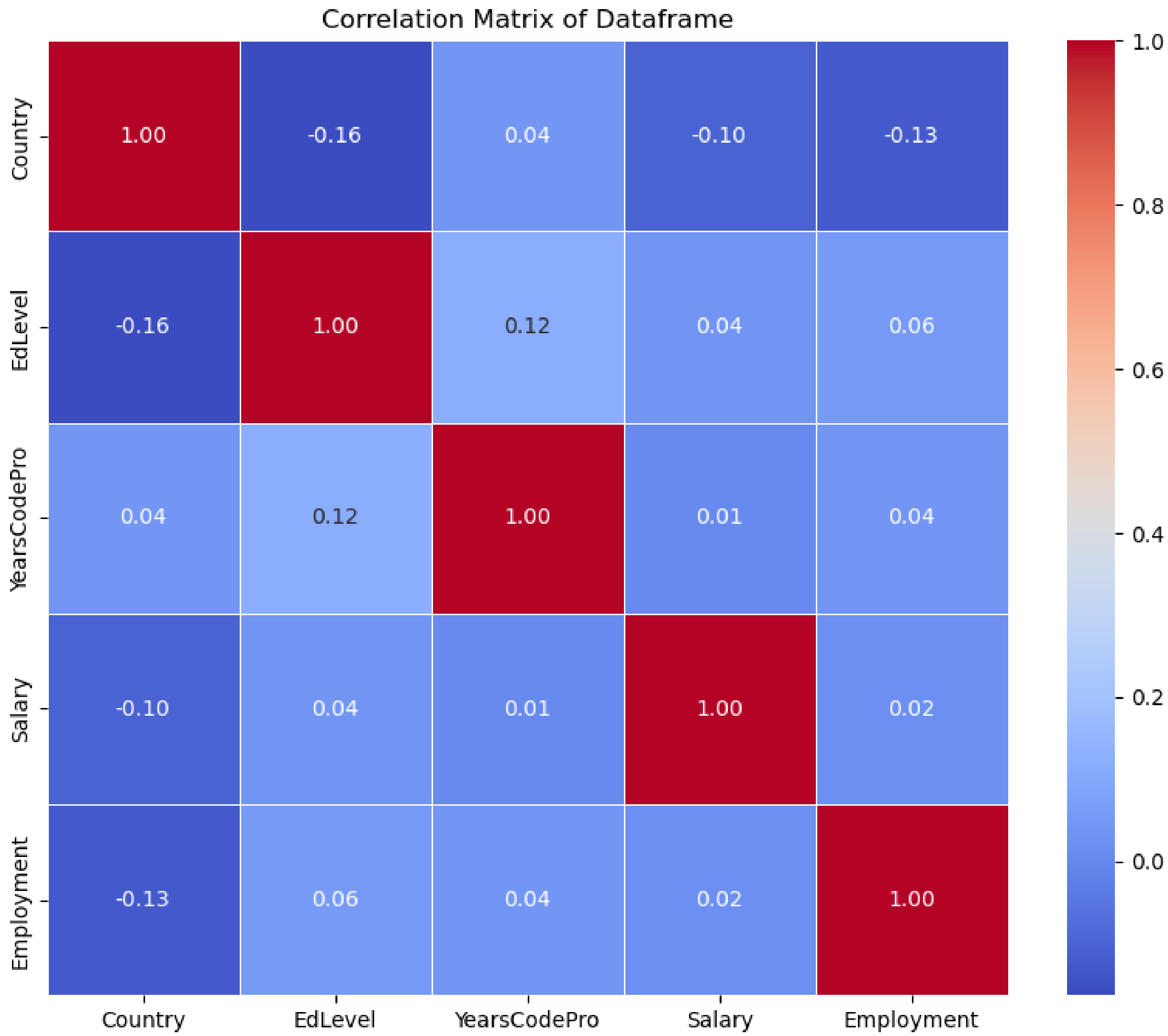
Each cell in the matrix represents the correlation coefficient between two variables. A positive correlation (closer to 1) indicates a positive relationship, while a negative correlation (closer to -1) indicates a negative relationship. This helps identify which features are strongly correlated, providing insights into potential multicollinearity.

In [100…

```
np.random.seed(42)
data = pd.DataFrame(np.random.randn(100, 5), columns=["Country", "EdLevel", "YearsCode

# Compute the correlation matrix
correlation_matrix = data.corr()

# Plot the correlation matrix using seaborn
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f", linewidths=0.5
plt.title('Correlation Matrix of Dataframe')
plt.show()
```



Histogram of Salary:

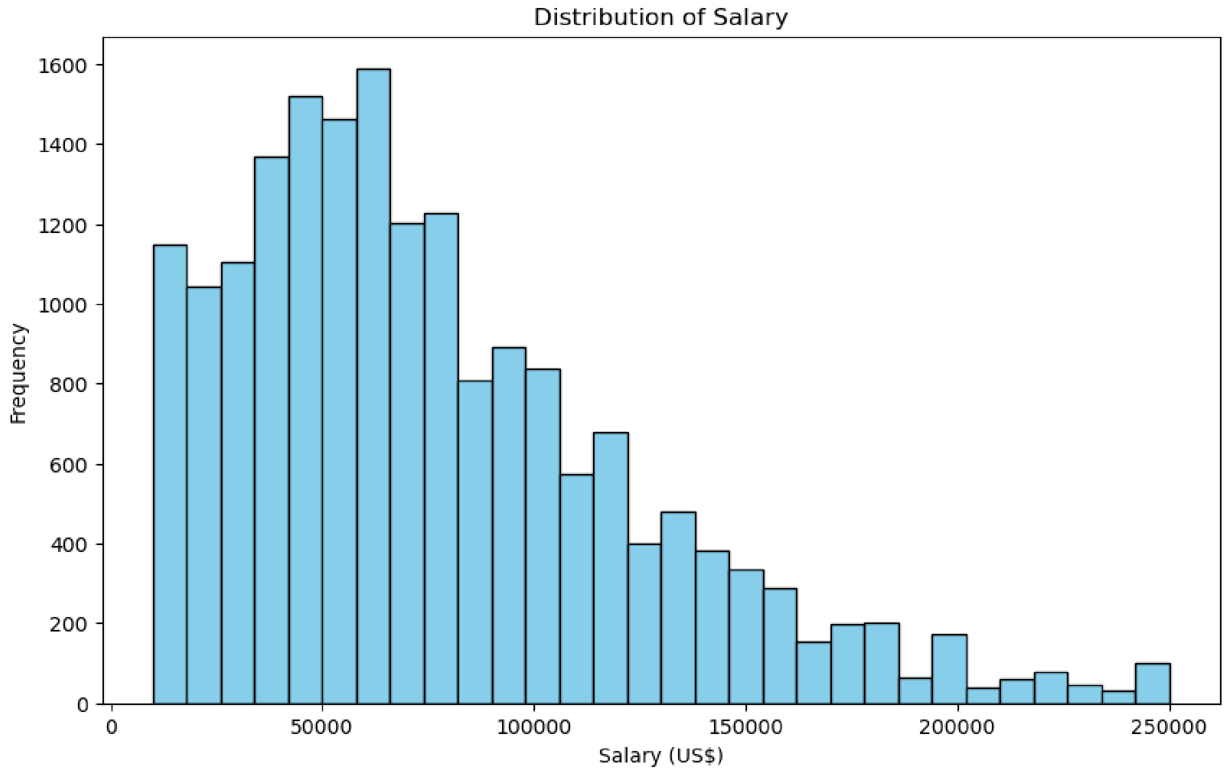
Purpose: The histogram provides a visual representation of the distribution of salaries, offering insights into the central tendency and variability of the salary data.

Histogram of Salary:

Purpose: The histogram provides a visual representation of the distribution of salaries, offering insights into the central tendency and variability of the salary data. Implementation:

In [100…

```
# Histogram of Salary
plt.figure(figsize=(10, 6))
plt.hist(df['Salary'], bins=30, color='skyblue', edgecolor='black')
plt.title('Distribution of Salary')
plt.xlabel('Salary (US$)')
plt.ylabel('Frequency')
plt.show()
df.head()
```



Out[1009]:

	Country	EdLevel	YearsCodePro	Salary
7	United States	Bachelor’s degree (B.A., B.S., B.Eng., etc.)	13	116000.0
9	United Kingdom	Master’s degree (M.A., M.S., M.Eng., MBA, etc.)	4	32315.0
10	United Kingdom	Bachelor’s degree (B.A., B.S., B.Eng., etc.)	2	40070.0
11	Spain	Some college/university study without earning ...	7	14268.0
12	Netherlands	Secondary school (e.g. American high school, G...	20	38916.0

In [101…

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

Out[1010]:

```
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)
```

In [101…

```
# Clean and transform the "YearsCodePro" column
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)
```

In [101…

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

Out[1012]:
```
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)
```

In [101…
```
# Clean and transform the "EdLevel" column
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)
```

In [101…
```
df["EdLevel"].unique()
```

Out[1014]:
```
array(['Bachelor’s degree', 'Master’s degree', 'Less than a Bachelors',
       'Post grad'], dtype=object)
```

In [101…
```
from sklearn.preprocessing import LabelEncoder
# Encode the categorical features using LabelEncoder
le_education = LabelEncoder()
df['EdLevel'] = le_education.fit_transform(df['EdLevel'])
df["EdLevel"].unique()
```

Out[1015]:
```
array([0, 2, 1, 3])
```

In [101…
```
le_country = LabelEncoder()
df['Country'] = le_country.fit_transform(df['Country'])
df["Country"].unique()
```

Out[1016]:
```
array([13, 12, 10,  7,  4,  2,  6,  1,  3,  5, 11,  8,  0,  9])
```

In [101…
```
# Split the dataset into features (X) and target variable (y)
X = df.drop("Salary", axis=1)
y = df["Salary"]
```

Model Selection:

• Consider trying different regression models like Linear Regression model, Decision Tree Regression model,Random Forest Regression model.Then calculating the RMSE to compare their performance to identify the most suitable model for salary prediction

In [101…
```
# Train a Linear Regression model
from sklearn.linear_model import LinearRegression
```

```
linear_reg = LinearRegression()
linear_reg.fit(X, y.values)
```

Out[1018]: LinearRegression()

In [101…

```
# Make predictions with the Linear Regression model
y_pred = linear_reg.predict(X)
```

In [102…

```
# Calculate the RMSE for Linear Regression

from sklearn.metrics import mean_squared_error, mean_absolute_error
import numpy as np
error = np.sqrt(mean_squared_error(y, y_pred))
```

In [102…

```
# Print the RMSE for Linear Regression
error
```

Out[1021]: 39274.75368318509

In [102…

```
# Train a Decision Tree Regression model
from sklearn.tree import DecisionTreeRegressor
dec_tree_reg = DecisionTreeRegressor(random_state=0)
dec_tree_reg.fit(X, y.values)
```

Out[1022]: DecisionTreeRegressor(random_state=0)

In [102…

```
# Make predictions with the Decision Tree Regression model

y_pred = dec_tree_reg.predict(X)
```

In [102…

```
# Calculate the RMSE for Decision Tree Regression

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

$29,414.94

In [102…

```
# Train a Random Forest Regression model
from sklearn.ensemble import RandomForestRegressor
random_forest_reg = RandomForestRegressor(random_state=0)
random_forest_reg.fit(X, y.values)
```

Out[1025]: RandomForestRegressor(random_state=0)

Feature Importance Plot for Random Forest Regression:

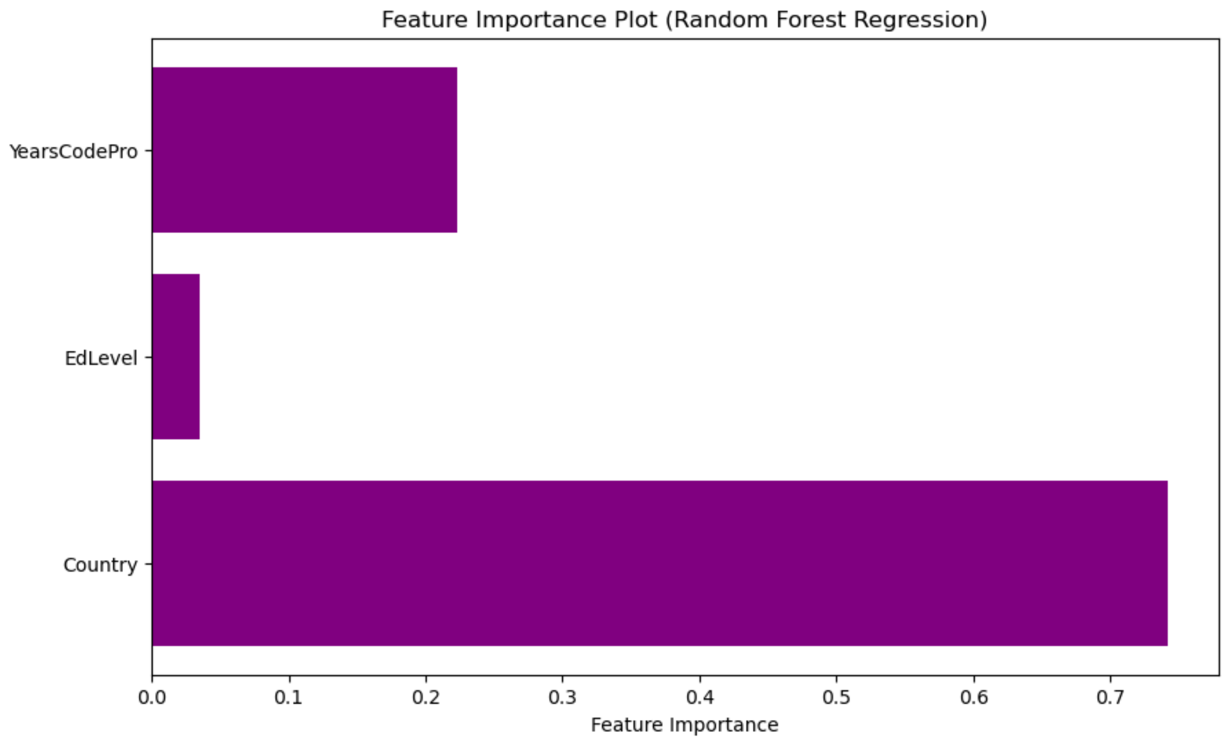
Purpose: Visualizing feature importance helps understand the contribution of different features in predicting the target variable (salary).

Explanation:

The horizontal bar plot displays the importance of each feature in predicting salaries. Higher bars indicate more influential features, helping prioritize which features have a greater impact on salary predictions. This information aids in feature selection and model interpretation.

In [102…

```
# Feature Importance Plot for Random Forest Regression
feature_importance = random_forest_reg.feature_importances_
features = X.columns
plt.figure(figsize=(10, 6))
plt.barh(features, feature_importance, color='purple')
plt.xlabel('Feature Importance')
plt.title('Feature Importance Plot (Random Forest Regression)')
plt.show()
```



These visualizations contribute to the exploratory data analysis (EDA) by providing a visual summary of key aspects of the dataset and relationships between variables. They enhance the understanding of salary distributions, variable relationships, and the importance of features in the context of predicting developer salaries.

In [102…

```
# Make predictions with the Random Forest Regression model
y_pred = random_forest_reg.predict(X)
```

In [102…

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

```
$29,487.31
```

In [102…

```
# Hyperparameter tuning for Decision Tree Regression using GridSearchCV
max_depth = [None, 2, 4, 6, 8, 10, 12]
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)
```

Out[1029]:

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

In [103…

```
# Get the best estimator from GridSearchCV
regressor = gs.best_estimator_
# Train the best Decision Tree Regression model
regressor.fit(X, y.values)
# Make predictions with the best Decision Tree Regression model
y_pred = regressor.predict(X)
# Calculate the RMSE for the best Decision Tree Regression model
error = np.sqrt(mean_squared_error(y, y_pred))
print("${:,.02f}".format(error))
```

```
$30,428.51
```

In [103…

```
X
```

Out[1031]:

	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 × 3 columns

In [103…

```
# Sample input for prediction
X = np.array([["United States", 'Master’s degree', 15 ]])
X
```

Out[1032]:

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

In [103…

```
X[:, 0] = le_country.transform(X[:,0])
X[:, 1] = le_education.transform(X[:,1])
X = X.astype(float)
X
```

Out[1033]:

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

In [103…
```
# Make a salary prediction for the sample input using the best Decision Tree Regressic
y_pred = regressor.predict(X)
y_pred
```

```
C:\Users\anishm\Anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X does
not have valid feature names, but DecisionTreeRegressor was fitted with feature names
  warnings.warn(
```

Out[1034]:
```
array([139427.26315789])
```

Evaluating the performance of the machine learning models is crucial to understand how well they are predicting salaries based on the given features. In this analysis, we have used three regression models: Linear Regression, Decision Tree Regression, and Random Forest Regression. The evaluation metric used here is the Root Mean Squared Error (RMSE), a commonly used metric for regression tasks.

Linear Regression Model:

The Linear Regression model is trained on the features (X) and target variable (y). Predictions (y_pred_linear) are made using the trained model. RMSE are calculated to evaluate the model's performance.

Decision Tree Regression Model:

The Decision Tree Regression model is trained on the features (X) and target variable (y). Predictions (y_pred_dt) are made using the trained model. RMSE are calculated for evaluation.

Random Forest Regression Model:

The Random Forest Regression model is trained on the features (X) and target variable (y). Predictions (y_pred_rf) are made using the trained model. RMSE are calculated for evaluation.

Model Comparison:

Now, you can compare the RMSE values for each model to determine their performance. Lower RMSE values indicate better model performance. Additionally, it's essential to consider other factors like interpretability, computational efficiency, and the specific goals of the analysis when choosing the best model.

Conclusion:

After evaluating the models, consider choosing the one with the lowest RMSE values. However, it's crucial to validate the model on unseen data (e.g., using a validation set or cross-validation) to ensure its generalization performance. This detailed evaluation allows you to understand how well each model is performing in predicting developer salaries based on the selected features.

In [103…

```
# Save the best model and encoding objects to a pickle file
import pickle
```

In [103…

```
data = {"model": regressor, "le_country": le_country, "le_education": le_education}
with open('saved_steps.pkl', 'wb') as file:
    pickle.dump(data, file)
```

In [103…

```
# Load the saved model and encoding objects from the pickle file
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"]
```

In [103…

```
# Make a salary prediction for the sample input using the loaded model
y_pred = regressor_loaded.predict(X)
y_pred
```

```
C:\Users\anishm\Anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X does
not have valid feature names, but DecisionTreeRegressor was fitted with feature names
  warnings.warn(
```

Out[1038]:

```
array([139427.26315789])
```

Detail summary of results and analysis

Data Preprocessing and Exploration: Initial Exploration:

Loaded the dataset, displayed the first few rows, and checked basic information using df.info(). Handling Missing Values:

Identified columns with missing values. Dropped rows where the target variable ("ConvertedComp") had missing values. Column Selection:

Selected specific columns of interest: "Country," "EdLevel," "YearsCodePro," "Employment," and "ConvertedComp." Renamed "ConvertedComp" to "Salary" for clarity. Filtering Employment Status:

Kept only rows where respondents are employed full-time. Dropped the "Employment" column as it was no longer needed. Country Data Transformation:

Grouped less frequent countries into an "Other" category for simplification. Further Data Filtering:

Removed outliers by restricting salary values to a range between $10,000 and 250$,000. Data Transformation - Years of Professional Coding:

Cleaned and transformed the "YearsCodePro" column to numerical values. Data Transformation - Education Level:

Cleaned and transformed the "EdLevel" column into categorical values. Applied label encoding to categorical features. Exploratory Data Analysis (EDA) Visualizations: Boxplot of Salary vs. Country:

Visualized the distribution of salaries across different countries using a boxplot. Identified variations in salary distributions and potential outliers for each country. Correlation Matrix:

Plotted a correlation matrix to understand the relationships between features. Explored how features are correlated with each other and with the target variable (Salary). Histogram of Salary:

Visualized the distribution of salaries using a histogram. Observed the frequency of different salary ranges among developers. Feature Importance Plot (Random Forest Regression):

Created a bar plot to visualize the importance of features in predicting salaries using Random Forest Regression. Model Training and Evaluation: Linear Regression Model:

Trained a Linear Regression model and evaluated it using RMSE. Decision Tree Regression Model:

Trained a Decision Tree Regression model and evaluated it using RMSE. Conducted hyperparameter tuning using GridSearchCV to improve the model. Random Forest Regression Model:

Trained a Random Forest Regression model and evaluated it using RMSE. Examined feature importance using a bar plot. Model Comparison: Compared the performance of all three models based on RMSE . Considered factors like interpretability, computational efficiency, and specific analysis goals. Conclusion and Recommendations: Model Selection:

Chose the model with the lowest RMSE values for predicting developer salaries. Insights:

Analyzed patterns in developer salaries across countries, considering various factors. Explored the impact of education level and years of professional coding experience on salaries. Recommendations:

Consider using the selected model for predicting salaries in similar contexts. Explore additional features or external factors that may influence salary predictions. This detailed analysis provides insights into the factors influencing developer salaries and helps in building a reliable predictive model. Continuous refinement and validation of the model on new data can further enhance its accuracy and applicability.

In [ ]:

```

```

In [ ]:

```

```