Software Developer Salary Prediction

By:
Chinnam Lakshmi Durga (RA1911003010127)
Akshitha Komatireddy (RA1911003010134)
Apurva Gaurav (RA1911003010140)

About the Project:
We have taken the dataset from StackOverFlow which is a survey of 65,000 software developers from 186 countries around the world on how they learn and level up, which tools they're using, and what they want. It examines all aspects of the developer experience from career satisfaction and job search to education and opinions on open Source Software.

Using the dataset, we have made a model that predicts a software developer's salary based on his work experience, the country he/she is belonging to and the educational qualification.
We have also made a user interactive application for the above to analyse a person's salary by providing the mentioned requirements above.

Dataset –
https://drive.google.com/file/d/1B-_4sZcAkYx8g4oufQWmEOh7kFh-7D80/view?usp=sharing (source:StackOverflow)

Complete code files –
https://github.com/ChinnamLakshmiDurga/Software-Developer-Salary-Prediction

Video Demonstration -
https://drive.google.com/file/d/1Bb4CjTTuxSu9LlKAaF0Wk9HN9JmQDOac/view?usp=sharing

Code Screenshots for the built Model :

```python
import pandas as pd
import matplotlib.pyplot as plt

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

```
Matplotlib is building the font cache; this may take a moment.
```

```python
df.head()
```

```
   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

show more (open the raw output data in a text editor) ...

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]
```

```python
#Cleaning the data
df = df[["Country", "EdLevel", "YearsCodePro", "Employment", "ConvertedComp"]]
df = df.rename({"ConvertedComp": "Salary"}, axis=1)
df.head()
```

```
              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
```

```
#Drop columns with NaN
df = df[df["Salary"].notnull()]
df.head()
```

```
           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
```

```
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
```

```
#Dropping rows where any of the columns is not a number
df = df.dropna()
df.isnull().sum()
```

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

```python
#Keeping only the data points where the employer is employed full time
df = df[df["Employment"] == "Employed full-time"]
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
```

```python
#Frequency of each country
df['Country'].value_counts()
```

```
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
```

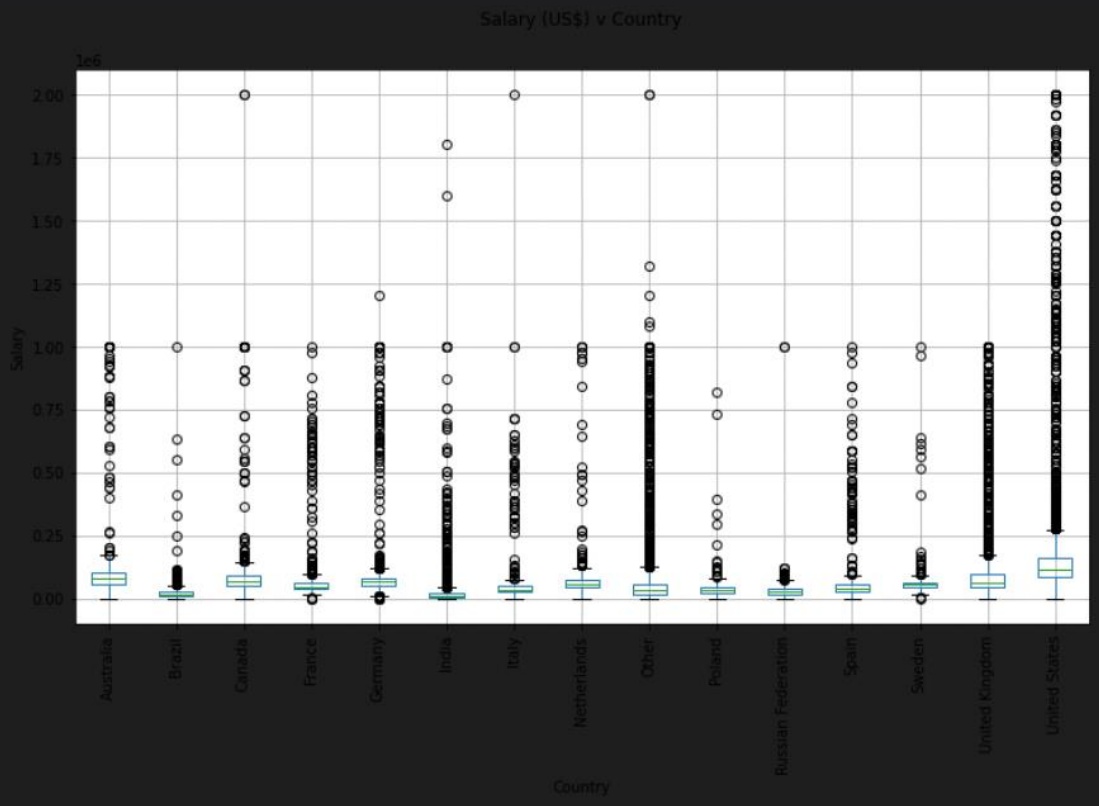
```python
#Getting rid of countries with few values
def shorten_categories(categories, cutoff):
    categorical_map = {}
    for i in range(len(categories)):
        if categories.values[i] >= cutoff:
            categorical_map[categories.index[i]] = categories.index[i]
        else:
            categorical_map[categories.index[i]] = 'Other'
    return categorical_map
```

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

```
Other                 8549
United States         7569
India                 2425
United Kingdom        2287
Germany               1903
Canada                1178
Brazil                 991
```

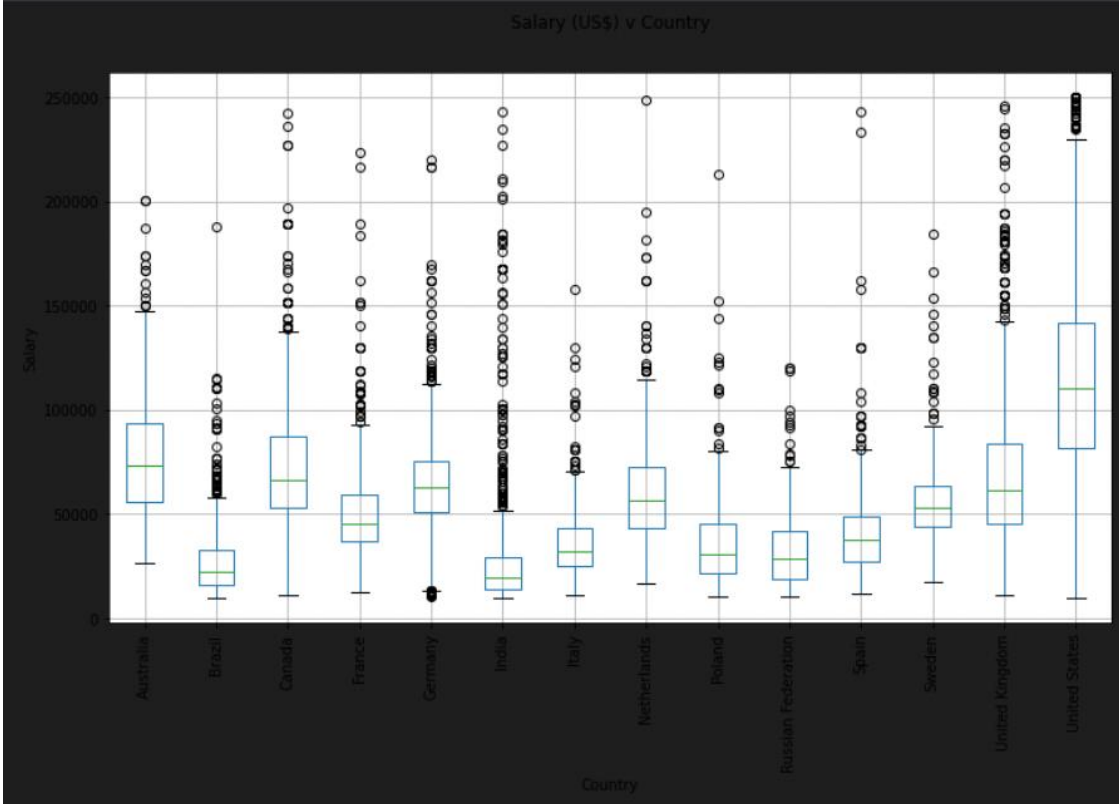
```
France                 972
Spain                  670
Australia              659
Netherlands            654
Poland                 566
Italy                  560
Russian Federation     522
Sweden                 514
Name: Country, dtype: int64
```

```
#Inspecting salary range -box Plot
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()
```




```
#Removing the outliers
df = df[df["Salary"] <= 250000]
df = df[df["Salary"] >= 10000]
#Removing other category
df = df[df['Country'] != 'Other']
```

```python
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()
```




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

```
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)
```

```python
def clean_experience(x):
    if x ==  'More than 50 years':
        return 50
    if x == 'Less than 1 year':
        return 0.5
    return float(x)
#Tranforming the data
df['YearsCodePro'] = df['YearsCodePro'].apply(clean_experience)
```

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

```
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)
```

```
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)
```

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

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

```
#Transforming string values to a integers
from sklearn.preprocessing import LabelEncoder
le_education = LabelEncoder()
df['EdLevel'] = le_education.fit_transform(df['EdLevel'])
df["EdLevel"].unique()
```

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

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

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

```
#Training Model
X = df.drop("Salary", axis=1) #All he columns except Salary.
y = df["Salary"]
```

```
from sklearn.linear_model import LinearRegression
linear_reg = LinearRegression()
#Fitting the data
linear_reg.fit(X, y.values)
```

```
LinearRegression()
```

```
#Prediction
y_pred = linear_reg.predict(X)
```

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

```
error
```

```
39274.75368318509
```

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

```
DecisionTreeRegressor(random_state=0)
```

```
#Prediction
y_pred = dec_tree_reg.predict(X)
```

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

```
$29,414.94
```

```
from sklearn.ensemble import RandomForestRegressor
random_forest_reg = RandomForestRegressor(random_state=0)
random_forest_reg.fit(X, y.values)
```

```
RandomForestRegressor(random_state=0)
```

```
#Prediction
y_pred = random_forest_reg.predict(X)
```

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

```
$29,487.31
```

```
#
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)
```

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

```
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
```

```
X
```

```
       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]
```

```
# country, edlevel, yearscode
X = np.array([["United States", 'Master's degree', 15 ]])
X
```

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

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

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

```
y_pred = regressor.predict(X)
y_pred
```

```
c:\users\dell\anaconda3\envs\dma\lib\site-packages\sklearn\base.py:445: UserWarning: X does not have valid feature names, but DecisionTreeRegressor was fitted with feature names
  warnings.warn(
array([139427.26315789])
```

+ Code + Markdown

```
#Saving the model
import pickle
```

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

```
#loading the model
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"]
```

```
y_pred = regressor_loaded.predict(X)
y_pred
```

```
c:\users\dell\anaconda3\envs\dma\lib\site-packages\sklearn\base.py:445: UserWarning: X does not have valid feature names, but DecisionTreeRegressor was fitted with feature names
  warnings.warn(
array([139427.26315789])
```

Output Screenshots:

Command Prompt - streamlit run app.py

```
Microsoft Windows [Version 10.0.19042.1237]
(c) Microsoft Corporation. All rights reserved.
The system cannot find the path specified.

C:\Users\Dell>conda activate DMA

(DMA) C:\Users\Dell>cd DMA

(DMA) C:\Users\Dell\DMA>streamlit run app.py

  You can now view your Streamlit app in your browser.

  Local URL: http://localhost:8501
  Network URL: http://192.168.29.123:8501
```

Explore Or Predict

Predict

Software Developer Salary Prediction

We need some information to predict the salary

Country

Australia

Education Level

Master's degree

Years of Experience

8

0 50

Calculate Salary

The estimated salary is $87522.38

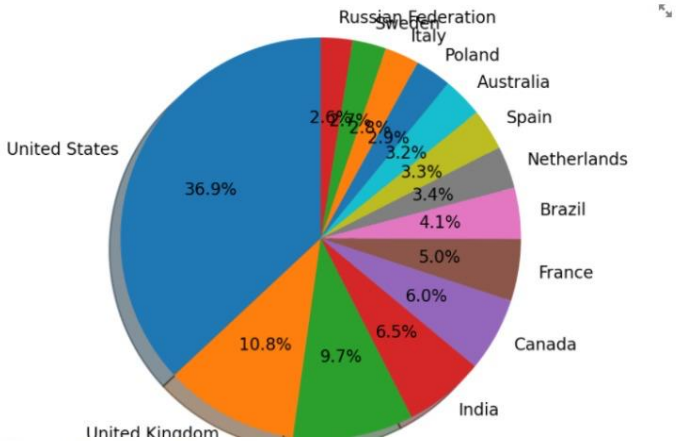
Explore Or Predict

Explore

Explore Software Engineer Salaries

Stack Overflow Developer Survey 2020

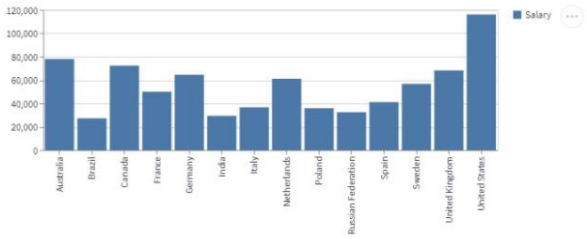
Number of Data from different countries



Explore Or Predict

Explore

Mean Salary Based On Country



Mean Salary Based On Experience