Machine Learning with Python

Data info:

Data Name: Data Science and STEM Salaries. Database source:
https://www.kaggle.com/datasets/jackogozaly/data-scienceand-stem-salaries
Data type: .CSV
Last Update: October 2021. Search/
Downloaded Date: 29, June 2022.
Rows:62643; Columns:29

Purpose: Use the data and the algorithms listed above to predict a base salary for Computer Science based in some variables, as years of experience, years at the company, race, gender and total yearly compensation.

Steps to read Dataset and necessary libraries.

1°: libraries necessary for the algorithms.
2°: reading data set.

In [ ]:
```
# importing necessary libraries
from google.colab import files
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression from
sklearn.neighbors import KNeighborsRegressor from
sklearn.metrics import mean_squared_error, r2_score from
sklearn import preprocessing
from sklearn.tree import DecisionTreeRegressor

import matplotlib.pyplot as plt
import plotly.express as px
import numpy as np import
seaborn as sb import pandas as
pd import io
```

```
# uploading the dataset to be read
```

```
# up = files.upload()

# read from cloud drive
df = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/DataScienceSa
laries.csv')

# print daset dimensions
print('\n Dimensions of dataset: Rows, Columns' , df.shape)
```

```
Dimensions of dataset: Rows, Columns (62642, 29)
```

In [ ]:

Steps to Data Cleaning

1°: removing unnecessary columns.
2°: check for NA’s, and remove them.
3°: boxplot to check outliers.
4°: check min and max values.
5°: set salaries higher than 700.000 to median value.
6°: set the years of experience equal median value.

In [ ]:

```
# remove unnecessary columns df = df.drop(columns=['timestamp',
 'otherdetails', 'cityid', 'dmaid',
 'rowNumber'])

 # print new dimensions print('\n New dimensions of dataset:
Rows, Columns', df.shape)
```

New dimensions of dataset: Rows, Columns (62642, 24)

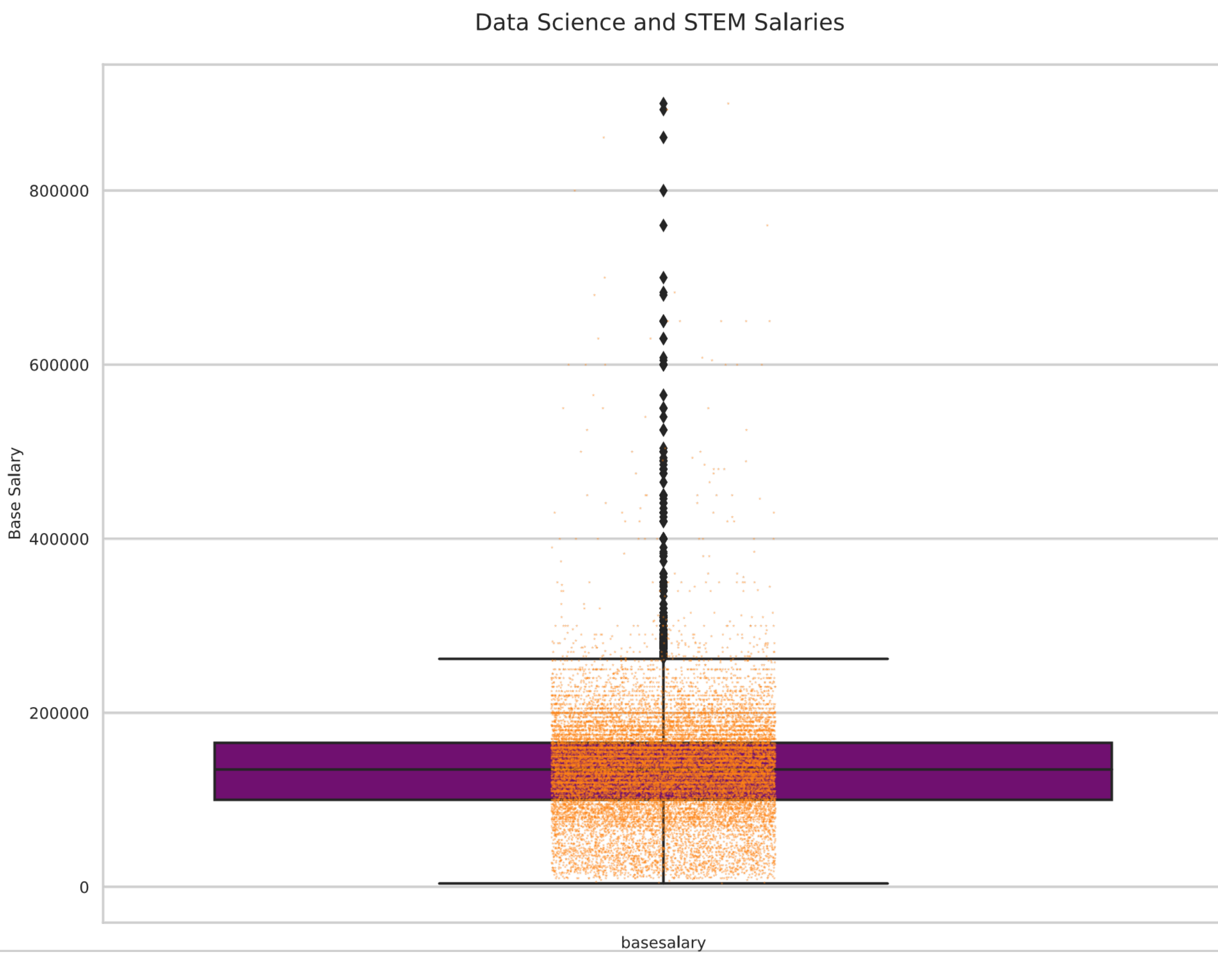
In [ ]:

```
# check for NA's
df.isnull().sum()

# remove NA's rows
df = df.dropna()
print('\n Dimensions after remove NA rows:' , df.shape)
```

 Dimensions after remove NA rows: (21521, 24)

In [ ]:

```
# 'base salary' boxplot to analyze mean and outliers
sb.set_theme(style="whitegrid")
bx_data = pd.DataFrame(data=df, columns=['basesalary'])
colors = ["purple"]
boxplot = sb.boxplot(x= "variable", y= "value", data= pd.melt(bx_data)
, order=['basesalary'], palette=colors)
boxplot = sb.stripplot(x= "variable", y= "value", data= pd.melt(bx_dat
a), alpha=.5, size=2.2,
          marker='*',  color="#FF8315", order=["basesalary"])

plt.title("Data Science and STEM Salaries \n", fontsize= 16)
plt.xlabel("-")
plt.ylabel("Base Salary", fontsize= 11)
plt.gcf().set_size_inches(14, 11)
plt.show()
```




```
# check for min and max base salary
min_sal = df['basesalary'].min()
max_sal = df['basesalary'].max()

print(f"Min base salary $ {min_sal:.2f}")
print(f"Max base salary $ {max_sal:.2f}")
```

In [ ]:

```
Min base salary $ 4000.00
Max base salary $ 900000.00
```

In [ ]:

```
# rows with info about min and max salary
print(df.loc[[df['basesalary'].idxmin()]], "\n\n\n")
print(df.loc[[df['basesalary'].idxmax()]])

# row with info about max year of experience
print(df.loc[[df['yearsofexperience'].idxmax()]])
```

```
                         company level             title  \
55840  Tata Consultancy Services    l1  Software Engineer

       totalyearlycompensation           location  yearsofexperience \
55840                    10000  Mumbai, MH, India                3.0

       yearsatcompany     tag  basesalary  stockgrantvalue  ...  \
55840             1.0  DevOps      4000.0           4000.0  ...

       Doctorate_Degree Highschool  Some_College  Race_Asian  Race_W
hite  \
55840                 0          0             0           1         0

       Race_Two_Or_More  Race_Black  Race_Hispanic   Race        Edu
cation
55840                 0           0              0  Asian  Master's
Degree

[1 rows x 24 columns]



      company               level                  title  \
45054     PwC  Partner / Principal  Management Consultant

       totalyearlycompensation     location  yearsofexperience  \
45054                   900000  Raleigh, NC               22.0

       yearsatcompany        tag  basesalary  stockgrantvalue  ...
\
```

```
45054              4.0  Cloud, IoT    900000.0              0.0  ...

       Doctorate_Degree Highschool  Some_College  Race_Asian  Race_W
hite  \
45054                 0          0             0           1         0

       Race_Two_Or_More  Race_Black  Race_Hispanic   Race        Edu
cation
45054                 0           0              0  Asian  Master's
Degree

[1 rows x 24 columns]
      company   level            title  totalyearlycompensation  \
39957     IBM  Band 9  Business Analyst                   155000

           location  yearsofexperience  yearsatcompany      tag  ba
sesalary  \
39957  San Jose, CA               45.0            20.0  Internal
155000.0

       stockgrantvalue  ...  Doctorate_Degree Highschool  Some_Colle
ge  \
39957              0.0  ...                 0          0             0

       Race_Asian  Race_White  Race_Two_Or_More  Race_Black  Race_Hi
spanic  \
39957           0           1                 0           0          0

        Race         Education
39957  White  Master's Degree

[1 rows x 24 columns]
```

In [ ]:

```
# seting salaries > 700k equal median salaries # set
years of experience greater then 30 equal median for
i, row in df.iterrows():
  df['basesalary'].values[df['basesalary'].values > 750000] = df['base
salary'].median()
  df['yearsofexperience'].values[df['yearsofexperience'].values > 35]
= df['yearsofexperience'].median()
```

Steps to Convert columns *(data cleaning)*

1°: making variables as factors.

In [ ]:
```
# converting columns as necessary df.Race
= df.Race.astype('category') df.gender =
df.gender.astype('category')
```

Steps to do Data Exploration *(data analysis)*

1°: some data analysis.
2°: analyzing min and max values

In [ ]:
```
# print the head of the data
df.head()
```

Out[ ]:

	company	level	title	totalyearlycompensation	location	yearsofexperience	years
15710	Google	L6	Software Engineer	400000	Sunnyvale, CA	5.0	
23532	Microsoft	61	Software Engineer	136000	Redmond, WA	3.0	
23533	Google	L5	Software Engineer	337000	San Bruno, CA	6.0	
23534	Microsoft	62	Software Engineer	222000	Seattle, WA	4.0	
23535	Blend	IC3	Software Engineer	187000	San Francisco, CA	5.0	

5 rows × 24 columns

```
# print a similar to summary about target and predictors df.loc[:,
         ['basesalary', 'totalyearlycompensation', 'yearsofexperience',
'yearsatcompany', 'gender', 'Race']
      ].describe(include='all')
```

Out[ ]:

basesalary	totalyearlycompensation	yearsofexperience	yearsatcompany	gender

In [ ]:

count	21521.000000	2.152100e+04	21521.000000	21521.000000	21521	2
unique	NaN	NaN	NaN	NaN	3	
top	NaN	NaN	NaN	NaN	Male	
freq	NaN	NaN	NaN	NaN	17556	1
mean	133732.633242	1.979472e+05	7.102458	2.706566	NaN	
std	56193.462943	1.331233e+05	5.784815	3.328219	NaN	
min	4000.000000	1.000000e+04	0.000000	0.000000	NaN	
25%	100000.000000	1.190000e+05	3.000000	0.000000	NaN	
50%	135000.000000	1.740000e+05	6.000000	2.000000	NaN	
75%	165000.000000	2.450000e+05	10.000000	4.000000	NaN	
max	700000.000000	4.980000e+06	35.000000	40.000000	NaN	

In [ ]:
```
# check the medin
df['basesalary'].median()
```

Out[ ]: 135000.0

In [ ]:
```
# variables type (equivalent to str without values)
print(df.dtypes)
```

```
company                        object
level                          object
title                          object
totalyearlycompensation         int64
location                       object
yearsofexperience             float64
yearsatcompany                float64
tag                            object
basesalary                    float64
stockgrantvalue               float64
bonus                         float64
gender                       category
Masters_Degree                  int64
Bachelors_Degree                int64
Doctorate_Degree                int64
Highschool                      int64
Some_College                    int64
Race_Asian                      int64
Race_White                      int64
Race_Two_Or_More                int64
Race_Black                      int64
```

```
Race_Hispanic                      int64
Race                            category
Education                         object
dtype: object
```

```python
# peinting columns names
for col in df.columns:
    print(col)
```

```
company level title
totalyearlycompensation
location
yearsofexperience
yearsatcompany tag
basesalary
stockgrantvalue bonus
gender Masters_Degree
Bachelors_Degree
Doctorate_Degree
Highschool
Some_College
Race_Asian
Race_White
Race_Two_Or_More
Race_Black
Race_Hispanic
Race
Education
```

In [ ]:

```python
# analyzing data variables min
df.min()
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:2: Futu
reWarning:

Dropping of nuisance columns in DataFrame reductions (with 'numeric_
only=None') is deprecated; in a future version this will raise TypeE
rror.  Select only valid columns before calling the reduction.
```

Out[ ]:

```
company                                  10x Genomics
level
title                                Business Analyst
totalyearlycompensation                         10000
location                          Aachen, NW, Germany
yearsofexperience                                 0.0
yearsatcompany                                    0.0
tag                                          #finance
basesalary                                     4000.0
stockgrantvalue                                   0.0
```

In [ ]:

```
bonus                                   0.0
Masters_Degree                            0
Bachelors_Degree                          0
Doctorate_Degree                          0
Highschool                                0
Some_College                              0
Race_Asian                                0
Race_White                                0
Race_Two_Or_More                          0
Race_Black                                0
Race_Hispanic                             0
Education                 Bachelor's Degree dtype:
object
```

```
# analyzing data variables max
df.max()
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:2: Futu
reWarning:

Dropping of nuisance columns in DataFrame reductions (with 'numeric_
only=None') is deprecated; in a future version this will raise TypeE
rror.  Select only valid columns before calling the reduction.
```

Out[ ]:

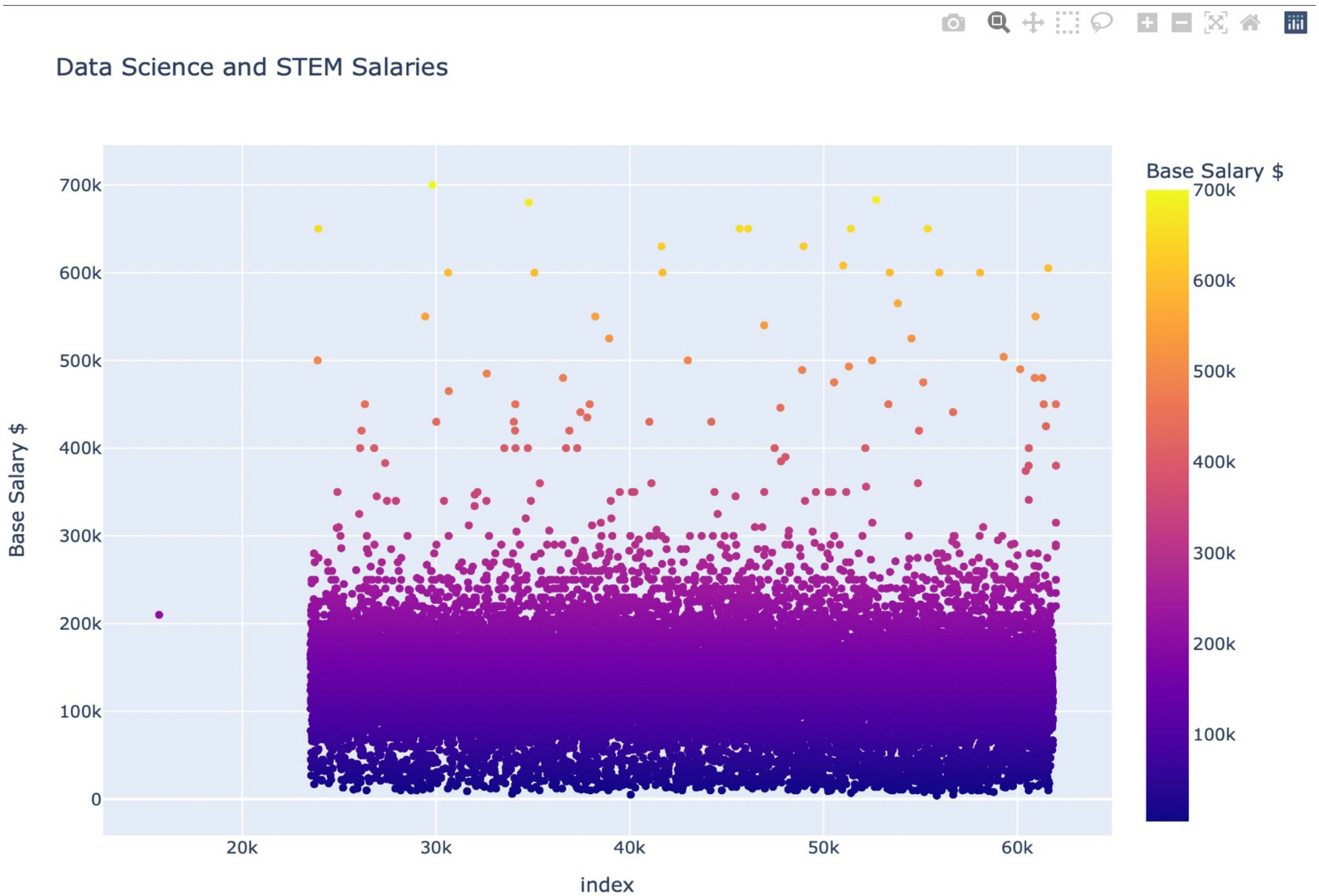
```
company                                          Google
level                                                專員
title                        Technical Program Manager
totalyearlycompensation                        4980000
location                      hod hasharon, HM, Israel
yearsofexperience                                 35.0
yearsatcompany                                    40.0
tag                                           whatsapp
basesalary                                    700000.0
stockgrantvalue                               954000.0
bonus                                         900000.0
Masters_Degree                                       1
Bachelors_Degree                                     1
Doctorate_Degree                                     1
Highschool                                           1
Some_College                                         1
Race_Asian                                           1
Race_White                                           1
Race_Two_Or_More                                     1
Race_Black                                           1
```

```
Race_Hispanic                                         1
Education                                             Some College dtype:
object
```

Steps to Data Exploration *(graphs)*

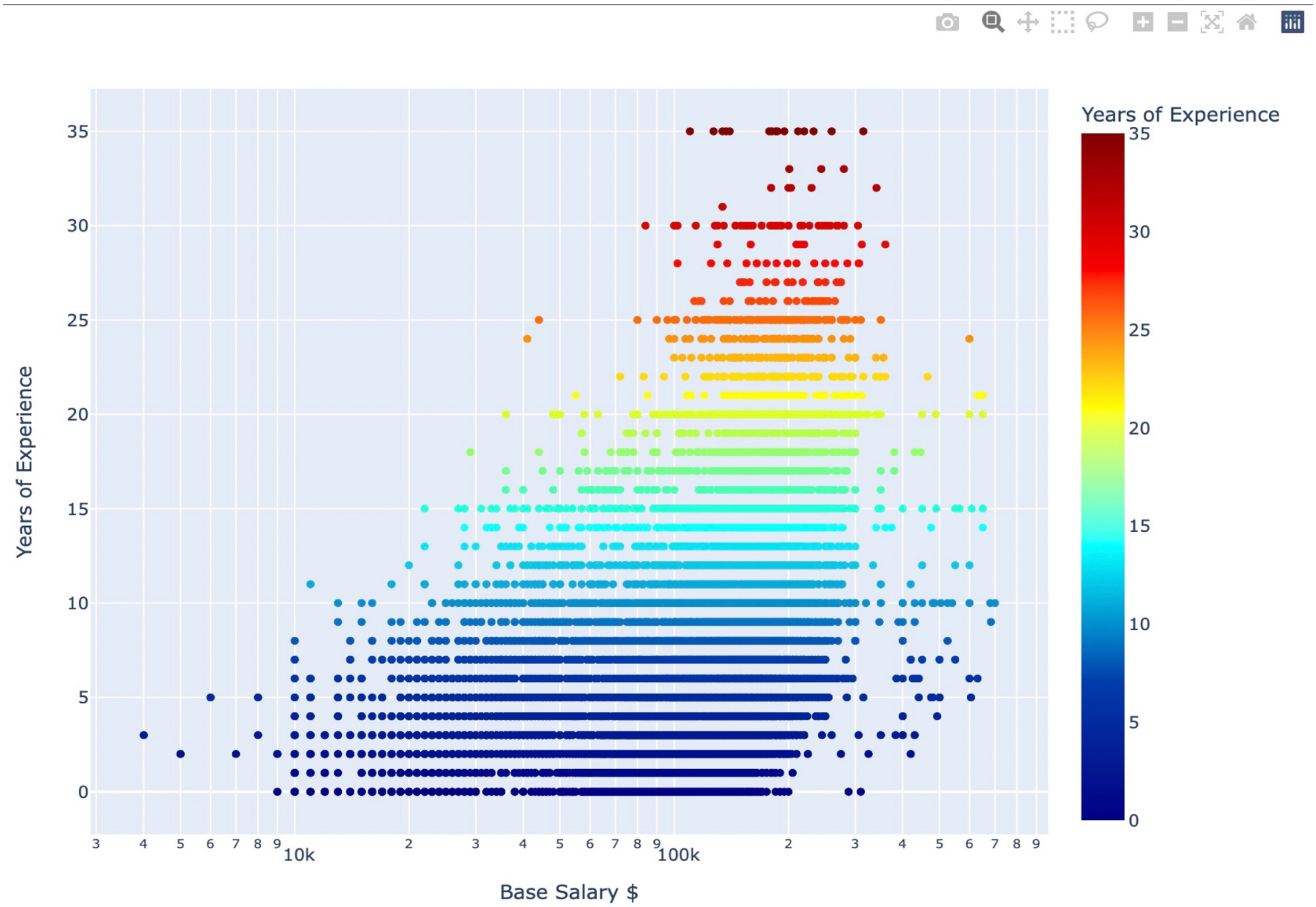
```
1°: graph to analyze base salaries and it range.
2°: graph analyzing base salary with years of experience.
3°: graph analyzing base salary with race.
```

In [ ]:
```
# Base Salary graph for analysis fig = px.scatter(df, y="basesalary", color="basesalary", width=930, he ight=675,
        labels={ "basesalary": "Base Salary $"}, title="Data Science and STEM Salaries")
fig.show()
```

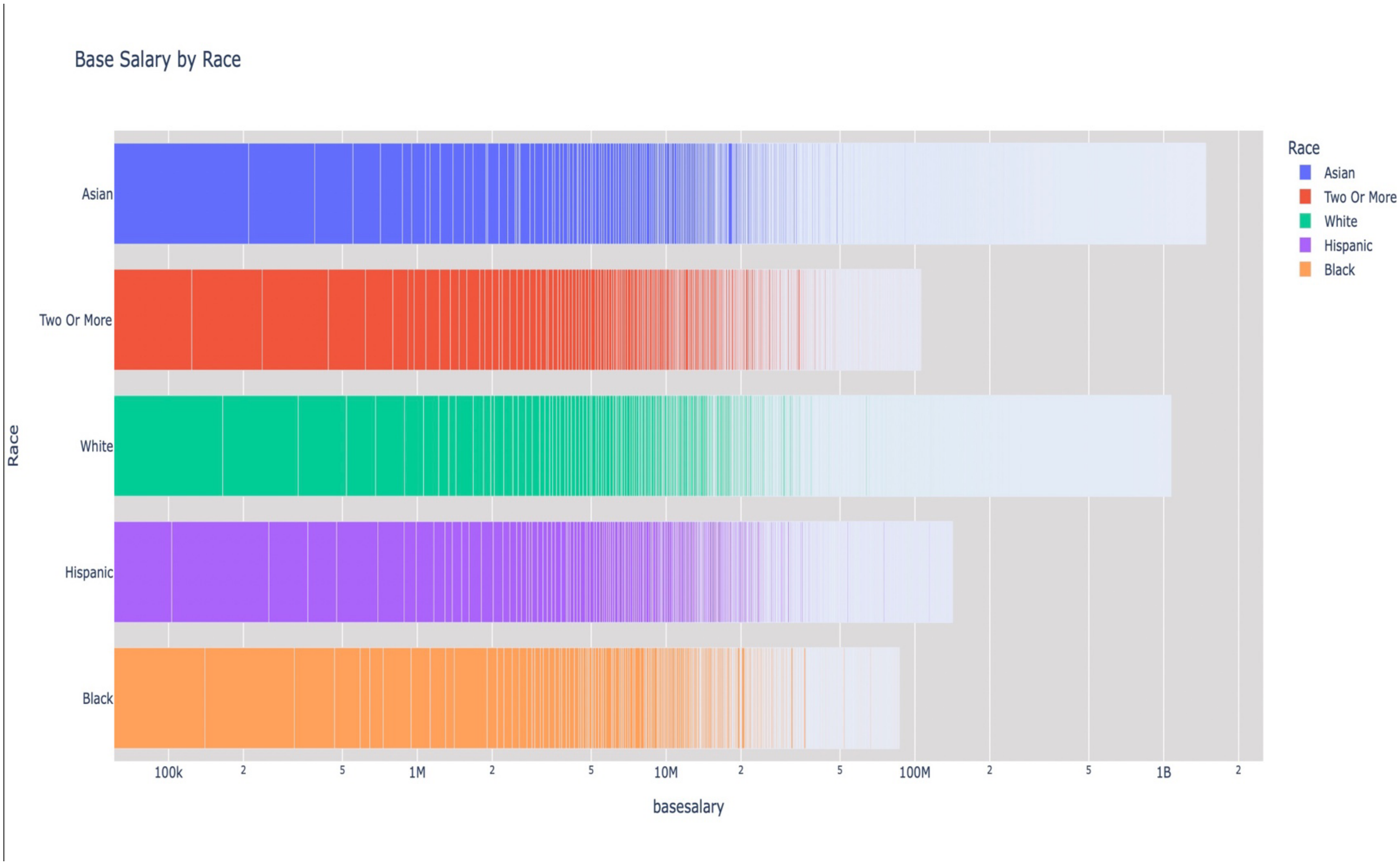


In [ ]:

```
# graph Base Salary vs Years of Experience
fig = px.scatter(df, x="basesalary", y="yearsofexperience", color='yea
rsofexperience', hover_name='company',
      width=930, height=675, color_continuous_scale=px.colors.sequenti
al.Jet,
      log_x= True, labels= dict(yearsofexperience= 'Years of Experienc
e', basesalary= 'Base Salary $')) fig.show()
```



```
# graph Base Salary vs Race (Asian, Black, Hispanic, White, Two or mor
e)
fig = px.bar(df, x="basesalary", y="Race", color='Race', orientation='
h', log_x=2,
             hover_data=["company", "company"], height=675, color_cont
inuous_scale='Bluered_r',
             title='Base Salary by Race')

fig.update_layout(plot_bgcolor='#DCDBDB')
fig.show()
```



Step to split data into Train and Test

```
1°: make a copy of the original dataser. 2°: refactor
Race and Gender arrays as integer values.
3°: dividing the data into 80% train and 20% test.
```

In [ ]:

```
# copy the dataset to keep original one
lr_df = df.copy()

# refactoring gender and Race array for int values lr_df['gender']
= (lr_df['gender'] == 'Female').astype(int)

lr_df['Race'] = lr_df['Race'].map({'Asian': 1,'White': 2, 'Black': 3,
'Hispanic': 4, 'Two Or More': 5})
```

In [ ]:

```
# dividing data into train and test x= lr_df.loc[:,
['totalyearlycompensation', 'yearsofexperience', 'year satcompany',
'gender', 'Race']] y= lr_df.basesalary

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size= 0
.2, random_state= 1234)

print("Train size:", x_train.shape)
print("Test size:", x_test.shape)
```

```
Train size: (17216, 5)
Test size: (4305, 5)
```

Step to make Linear Regression

1°: make a linear model with 5 predictors.
2°: calculate the model coefficient.
3°: make a model prediction.
4°: calculate the mse and correlation for the model.
5°: make a plot for correlation results.

In [ ]:

```
# make a linear model and train the data
lr = LinearRegression()
lr.fit(x_train, y_train)
```

Out[ ]: LinearRegression()

In [ ]:

```
# calculate coefficient
print("Intercept:", lr.intercept_)
print("Coefficient: ", lr.coef_)
```

```
Intercept: 61075.9163313023
```

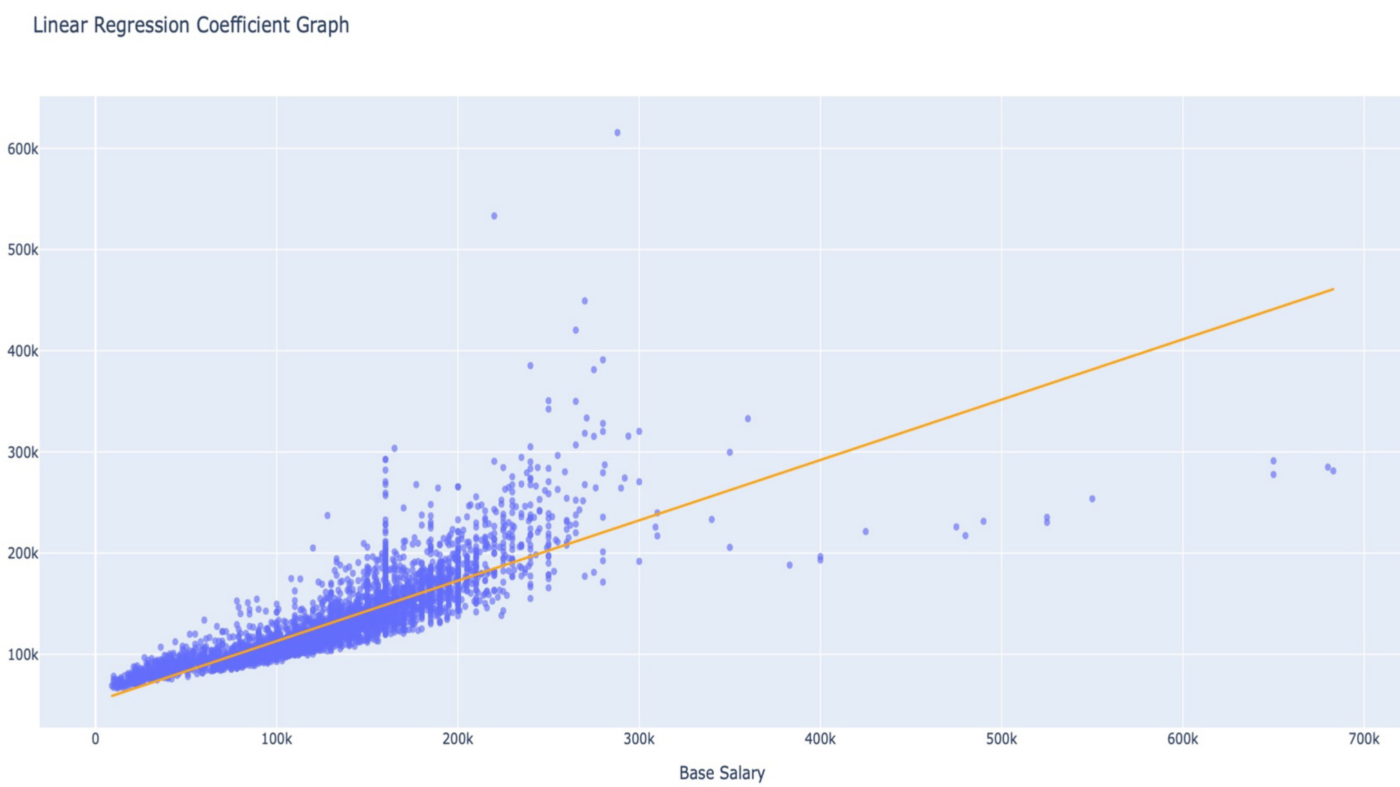
```
Coefficient:  [ 3.00237935e-01  1.39234603e+03 -2.27130377e+02  5.21
128968e+03
  1.73397000e+03]
```

In [ ]:
```
# make a model prediction lr_pred
= lr.predict(x_test)
```

In [ ]:
```
# printing linear regression and correlation results print("Linear
Regression MSE: ", mean_squared_error(y_test, lr_pred)) print("Linear
Regression Correlation: ", r2_score(y_test, lr_pred))
```

```
Linear Regression MSE:  1074363705.4151342
Linear Regression Correlation:  0.6619116564026211
```

In [ ]:
```
# correlation plot
fig = px.scatter(
    df, x=y_test, y=lr_pred, opacity=0.65, labels=dict(x='Base Salary'
, y=''),
    title='Linear Regression Coefficient Graph', height= 675,
trendline='ols', trendline_color_override='orange'
)
fig.show()
```



KNN Algorithm

Step to make KNN

```
1°: make a prediction using KNN regression algorithm
2°: make KNN prediction
3°: calculate MSE and correlation ofr the KNN model.
```

In [ ]:
```
# make a knn model with k= 3
knn = KNeighborsRegressor(n_neighbors=3)
knn.fit(x_train, y_train)
```

Out[ ]: `KNeighborsRegressor(n_neighbors=3)`

In [ ]:
```
# make a knn prediction kn_pred
= knn.predict(x_test)
```

In [ ]:
```
# printing knn mse and correlation results print("KNN
MSE:", mean_squared_error(y_test, kn_pred)) print("KNN
Correlation: ", r2_score(y_test, kn_pred))
```

```
KNN MSE: 820701174.3450768
KNN Correlation:  0.7417359696495549
```

Comparison: *Linear Regression vs KNN*

Linear Regression has MSE: 107 and Correlation: 0.662. KNN has a much higher MSE: 821 and Correlation: 0.742.

I tried K with different values to check for a better performance but k= 3 has the best values, since high k values increases MSE as well and does not increase Correlation.

Scale KNN

Steps to Scale KNN

```
1°: set up train and test scale.
2°: make a knn scale model. 3°:
make a knn scale prediction.
```

```
4°: calculate mse and correlation for scale.
5°: print and compare results.
```

In [ ]:
```
# make train and test scale
scale = preprocessing.StandardScaler().fit(x_train)

x_train_scale = scale.transform(x_train)
x_test_scale = scale.transform(x_test)
```

In [ ]:
```
# make a scale model
knn_sc = KNeighborsRegressor(n_neighbors= 3)
knn_sc.fit(x_train_scale, y_train)
```

Out[ ]:
```
KNeighborsRegressor(n_neighbors=3)
```

In [ ]:
```
# make a knn scale prediction kn_scale_pred =
knn_sc.predict(x_test_scale)
```

In [ ]:
```
# calculate and print scale mse and correlation print("Scale
MSE: ", mean_squared_error(y_test, kn_scale_pred)) print("Scale
Correlation_: ", r2_score(y_test, kn_scale_pred))
```

```
Scale MSE:  822754419.9251516
Scale Correlation_:  0.7410898398578771
```

Comparison: *Linear Regression vs KNN vs Scale*

LinReg - MSE: 107 | Correlation: 0.662.

KNN - MSE: 821 | Correlation: 0.742.

Scale - MSE:823 | Correlation: 0.741

Comparing all tree algorithms, we can see that Linear Regression has the worst result for Correlation but the best for MSE, and KNN Scale has a little improvement in MSE but on the other hand decrease very little on Correlation. Comparing against KNN since run time wasn't used, we can say that KNN and Scale has the same result or are the best algorithm for this situation.

Decision Tree

Steps to do Decision Tree

1°: make a prediction using 5 predictors.
2°: plot the prediction to analyze the graph.
3°: make a prediction, calculate correlation and mse.

In [ ]:
```
# make a decision tree model
dt = DecisionTreeRegressor()
dt.fit(x_train, y_train)
```

Out[ ]: DecisionTreeRegressor()

In [ ]:
```
# make decision tree prediction dt_pred
= dt.predict(x_test)
```

In [ ]:
```
# calculate and print DT mse and correlation print("Decision Tree
MSE: ", mean_squared_error(y_test, dt_pred)) print("Decision Tree
Correlation_: ", r2_score(y_test, dt_pred))
```

```
Decision Tree MSE:  1285679853.7113464
Decision Tree Correlation_:  0.5954132013703572
```

Conclusion.

Comparison: *Linear Regression vs KNN vs Scale vs Decision Tree*

LinReg - MSE: 107 | Correlation: 0.662.

KNN - MSE: 821 | Correlation: 0.742.

Scale - MSE: 823 | Correlation: 0.741

DecTree - MSE:128 | Correlation: 0.597

The final algorithm did no bit the KNN and Scale KNN, the Decision Tree performed little bit better then Linear Regression only.

So, taking into a count that a good algorithm should give the higher correlation and lower MSE.

In this case we could conclude that Linear Regression performed best since the MSE is way lower than the others, and its Correlation is not lower by far. The other algorithms have a good value for Correlation, but a MSE too higher.

Comparing to the R-project, the results does not match, because the best results were in KNN for when k= 15, and Decision Tree both with results not equal but very closer.

UTD, Texas Dallas.

Introduction to Machine Learning (CS-4575)
Instructor: Dr. Mazidi Celio L.

Converting file to PDF.

In [ ]:
```
# code to generate a PDF file
!jupyter nbconvert --to pdf /content/machLearn-Python.ipynb
```

```
[NbConvertApp] Converting notebook /content/machLearn-Python.ipynb t o
pdf
/usr/local/lib/python3.7/dist-packages/nbconvert/filters/datatypefil
ter.py:41: UserWarning: Your element with mimetype(s) dict_keys(['te
xt/html']) is not able to be represented.
  mimetypes=output.keys())
/usr/local/lib/python3.7/dist-packages/nbconvert/filters/datatypefil
ter.py:41: UserWarning: Your element with mimetype(s) dict_keys(['te
xt/html']) is not able to be represented.
  mimetypes=output.keys())
[NbConvertApp] Support files will be in machLearn-Python_files/
[NbConvertApp] Making directory ./machLearn-Python_files
[NbConvertApp] Making directory ./machLearn-Python_files
[NbConvertApp] Writing 71911 bytes to ./notebook.tex
[NbConvertApp] Building PDF
[NbConvertApp] Running xelatex 3 times: ['xelatex', './notebook.tex' ,
'-quiet']
[NbConvertApp] Running bibtex 1 time: ['bibtex', './notebook']
[NbConvertApp] WARNING | bibtex had problems, most likely because th
ere were no citations
[NbConvertApp] PDF successfully created
[NbConvertApp] Writing 317821 bytes to /content/machLearn-Python.pdf
```