Exploring Different Machine Learning Algorithms

BY MICHELLE REYES, MEGAN BEE, MICHAEL CASTILLO, BRITNEY COLLIER, NICHOLAS HOANG, HANMO ZHANG, GABRIEL ROBLES, BIJOU RAJ, AND JOHNNY GARCIA.

DESCRIPTION

We are exploring different machine learning algorithms. Our primary objective is to address one problem: predicting an individual's salary based on a set of variables. We will then analyze, compare, and contrast the models and understand which model is the most effective.

GOALS

Learn how to implement different algorithms to a model.

Learn an outline that is applicable to most machine learning models.

Learn how to identify suitable algorithms and seeking the most efficient solution for the dataset.

Technologies

Machine Learning Libraries

Sklearn

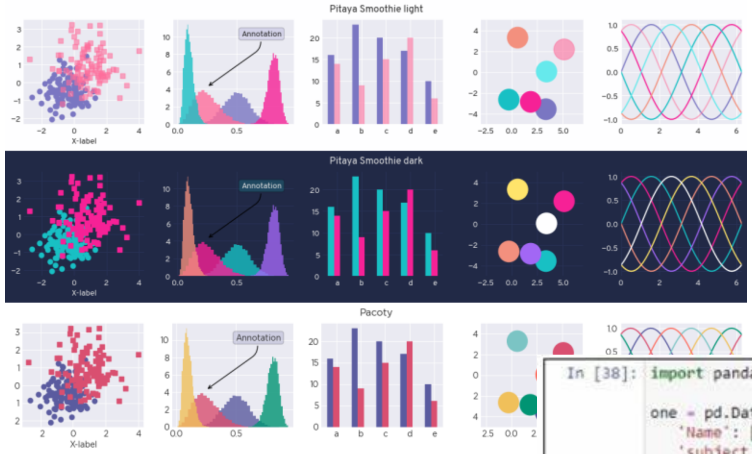
Tensorflow

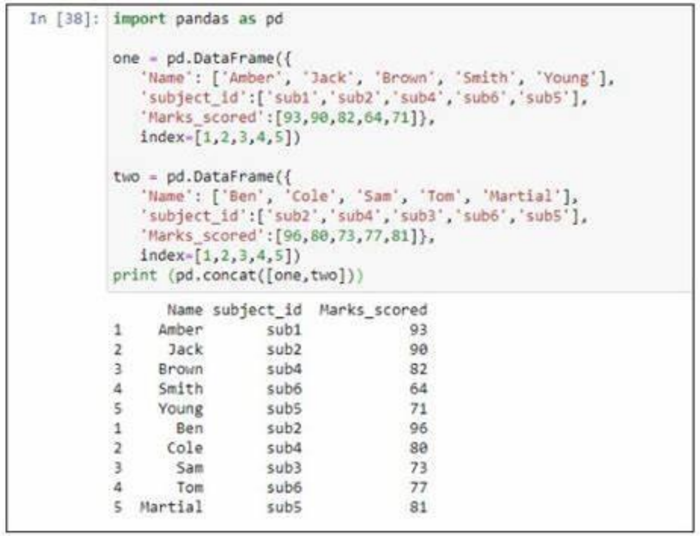
Python Libraries

- Seaborn
- Matplotlib
- Pandas
- Numpy

Code Editors:

- VsCode/Jupyter Notebook





```
In [38]: import pandas as pd

         one = pd.DataFrame({
           'Name': ['Amber', 'Jack', 'Brown', 'Smith', 'Young'],
           'subject_id':['sub1','sub2','sub4','sub6','sub5'],
           'Marks_scored':[93,90,82,64,71]},
           index=[1,2,3,4,5])

         two = pd.DataFrame({
           'Name': ['Ben', 'Cole', 'Sam', 'Tom', 'Martial'],
           'subject_id':['sub2','sub4','sub3','sub6','sub5'],
           'Marks_scored':[96,80,73,77,81]},
           index=[1,2,3,4,5])
         print (pd.concat([one,two]))

      Name subject_id  Marks_scored
1    Amber       sub1            93
2     Jack       sub2            90
3    Brown       sub4            82
4    Smith       sub6            64
5    Young       sub5            71
1      Ben       sub2            96
2     Cole       sub4            80
3      Sam       sub3            73
4      Tom       sub6            77
5  Martial       sub5            81
```

STEP 1: How we picked our algorithms?

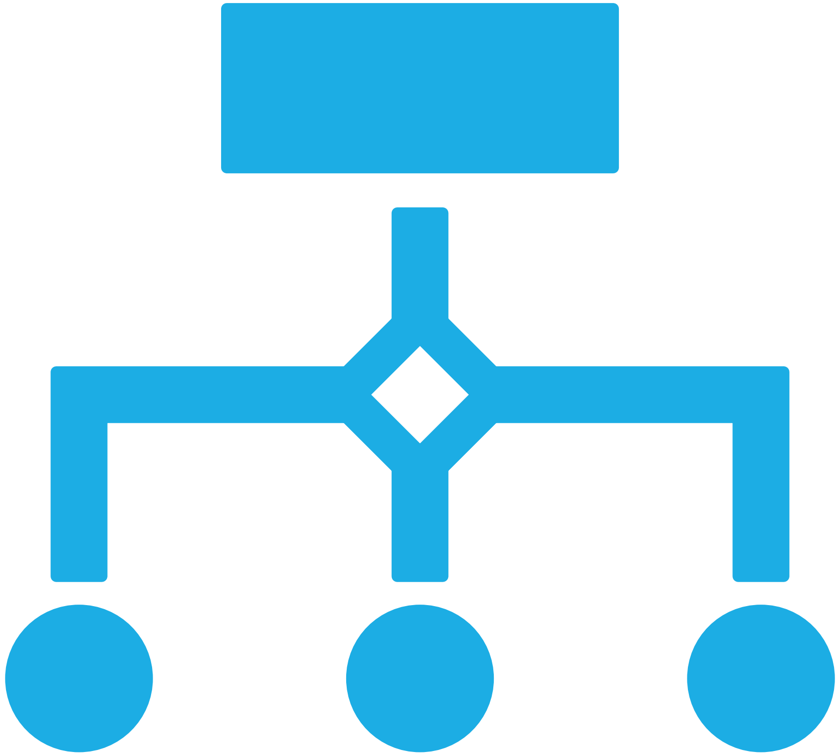
Given our dataset is labeled, we decided **supervised learning** is the machine learning technique we will using. Since we are predicting someone's salary, we know we are dealing with a regression problem.

As result, we picked only algorithms that deal with regression problems and part of supervised learning:

- K-nearest Neighbors
- Neural Networks
- Decision Tree
- Random Forest
- Linear Regression
- SVM
- XGBoost

SALARY PREDICTOR

# Age	Gender	Education ...	Job Title	# Years of E...	# Salary
32	Male	Bachelor's	Software Engineer	5	90000
28	Female	Master's	Data Analyst	3	65000
45	Male	PhD	Senior Manager	15	150000
36	Female	Bachelor's	Sales Associate	7	60000
52	Male	Master's	Director	20	200000
29	Male	Bachelor's	Marketing Analyst	2	55000
42	Female	Master's	Product Manager	12	120000
31	Male	Bachelor's	Sales Manager	4	80000
26	Female	Bachelor's	Marketing Coordinator	1	45000
38	Male	PhD	Senior Scientist	10	110000
29	Male	Master's	Software Developer	3	75000
48	Female	Bachelor's	HR Manager	18	140000

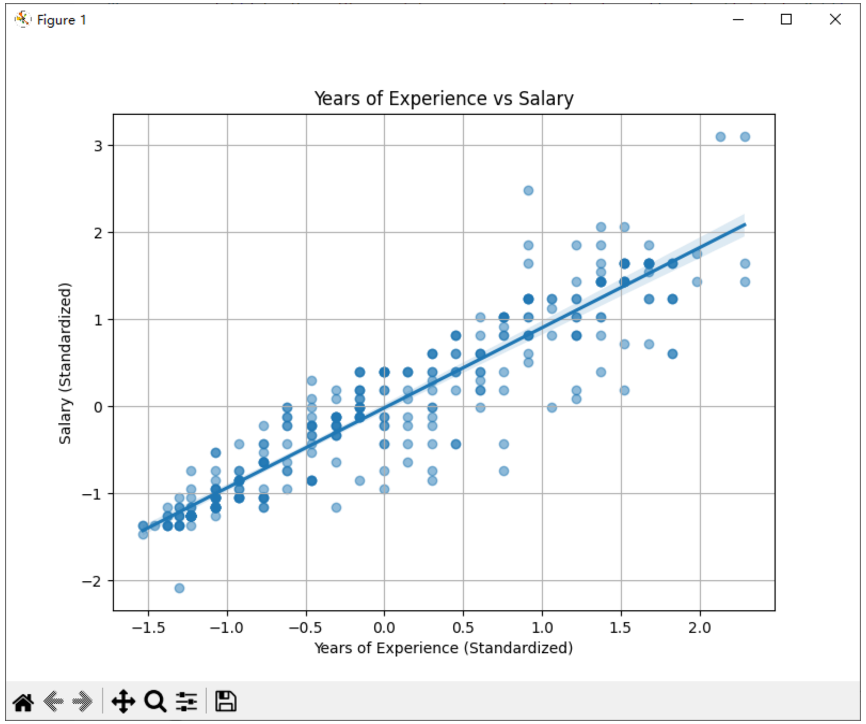


STEP 2: Model Implementation

Small sub-groups tackled each Algorithm.

Support Vector Machine

- Start with low-dimensional data.
- Move the data to a higher dimension.
- Find a Support Vector Classifier to distinguish between the two groups of data.



Support Vector Machine

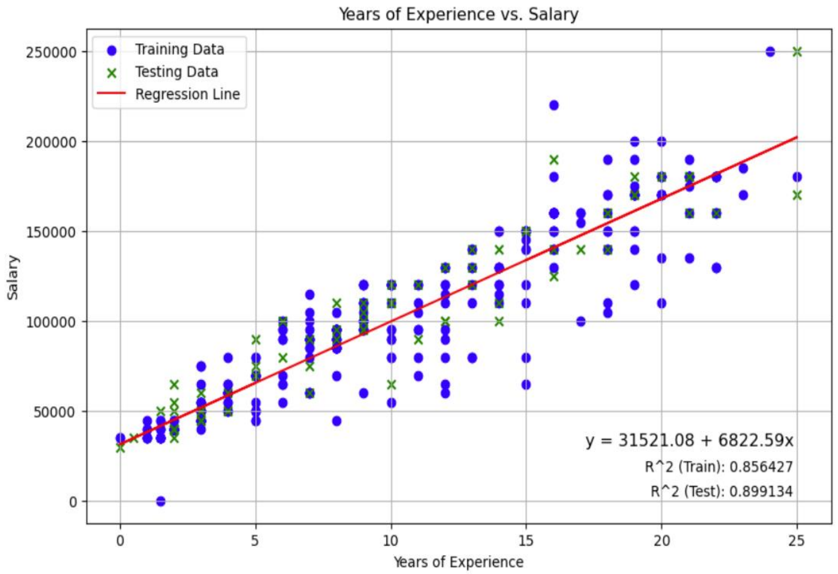
One attribute in the data set is Job title which is difficult to quantify. So I used ***Bert*** model tokenize the attribute so the model can understand the context of each job title.

```
        Age  Years of Experience    Salary         0         1  ...  Gender_nan  Education Level_Bachelor's  Education Level_Master's  Education Level_PhD  Education Level_nan
0 -0.769398            -0.768276 -0.219559  0.716585 -0.023704  ...   -0.073225                    0.821040                 -0.594803            -0.396746            -0.073225
1 -1.336003            -1.073702 -0.738498  0.742144 -1.425594  ...   -0.073225                   -1.217967                  1.681229            -0.396746            -0.073225
2  1.072068             0.758859  1.025892 -0.510377  1.501051  ...   -0.073225                   -1.217967                 -0.594803             2.520504            -0.073225
3 -0.202793            -0.462849 -0.842285 -0.096239  0.678115  ...   -0.073225                    0.821040                 -0.594803            -0.396746            -0.073225
4  2.063627             1.522426  2.063768  0.165148  0.849588  ...   -0.073225                   -1.217967                  1.681229            -0.396746            -0.073225
```

```
PS E:\machine_learning> & C:/Python311/python.exe e:/machine_lear
Best Parameters: {'C': 100, 'gamma': 'auto', 'kernel': 'rbf'}
Best cross-validation MSE: -0.14232745701101757
Mean Squared Error: 0.05711448664234943
R^2 Score: 0.881541796993576
        Actual      Predicted
114    35000.0   34239.584477
296   130000.0  111637.461521
231   120000.0  134058.378946
322   130000.0  121381.895836
137    50000.0   57779.104159
PS E:\machine_learning>
```

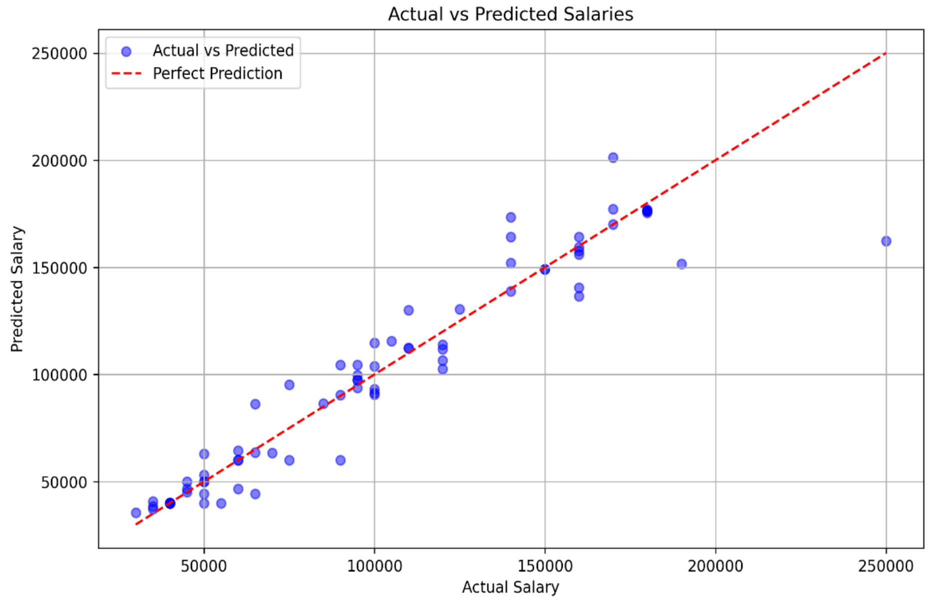
Linear Regression

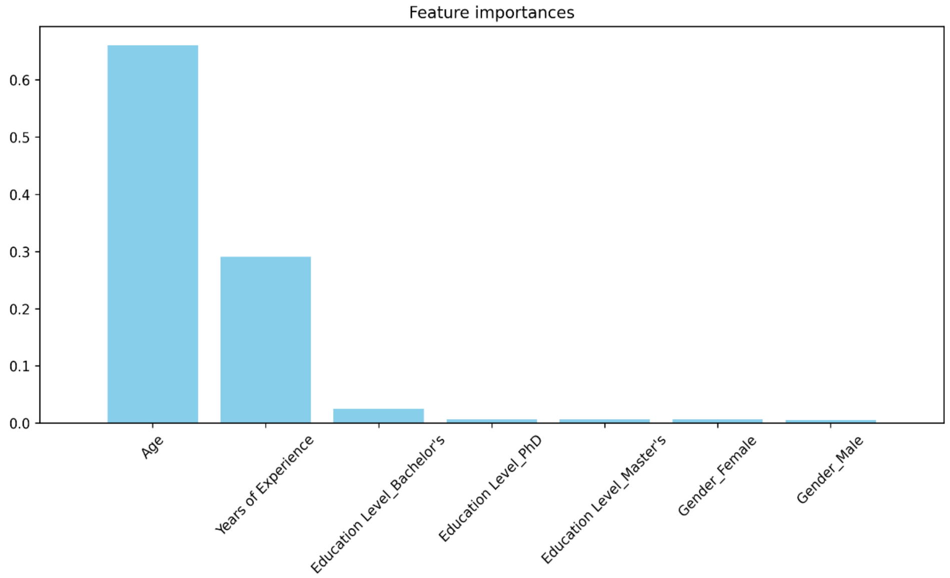
- First put all data into model, then 80/20 split
- Testing r^2 values were 0.899 and 0.849 for years of experience and age
- Linear regression ideal for data set, but not perfect
- Learned the importance of data splitting in models



Random Forest

- 80/20 Training/testing Split
- 100 Decision Trees
- Accuracy: 0.904305653647498
- Feature Importances
  - Age
  - Experience
  - Bachelor's Degree





Neural Networks

- Omitted Job Title column due to difficulty in categorizing numerous range of values
- Created training and validation sets with a 90/10 split
- Input data were Age, Gender, Education Level, and Years of Experience
- Decided to use ReLu Activation function to speed up and simplify model training
- Output node held the salary prediction

- Changes to our model: the number of layers, types of layers, and nodes within each layer
- Training mean absolute error: 15936.1221
- Validation mean absolute error: 11246.9131
- Train R^2 value: 0.98854
- Test R^2 value: 0.88547
  - A 0.10 difference in R^2 values could indicate a slight chance of overfitting

Predicted		Actual
[51074.297]	137	50000.0
[169398.19]	249	170000.0
[148866.38]	338	150000.0
[55787.055]	247	50000.0
[37595.723]]	97	35000.0

```
# Neural Network Architecture
model = keras.Sequential([
    keras.layers.Dense(128, activation='relu',
                       input_shape=(XtrainScaled.shape[1],)),
    keras.layers.Dropout(0.2),
    keras.layers.Dense(64, activation='relu'),
    keras.layers.Dropout(0.2),
    keras.layers.Dense(32, activation='relu'),
    keras.layers.Dropout(0.2),
    keras.layers.Dense(16, activation='relu'),
    keras.layers.Dense(1)  # No activation for output layer
])
```

```
Epoch 23/25
42/42 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 433505376.0000 - mae: 14858.6221 - val_loss: 245892816.0000 - val_mae: 122
7051
Epoch 24/25
42/42 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 472196160.0000 - mae: 15912.9395 - val_loss: 224921648.0000 - val_mae: 116
6846
Epoch 25/25
42/42 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 441410688.0000 - mae: 15936.1221 - val_loss: 212928800.0000 - val_mae: 112
9131
```

$$R^2(y,\hat{y}) = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

```
r2Value = 1-(squaredError/newDF.sum())
print(r2Value)
```

```
0.8854771279155983
```

K – Nearest Neighbors

The highest accuracy percentage achieved from this model was just under **50%**

Model Implementation:

- Started with 80/20 split
- Choose best K value (11)
- Train and test data

Model Improvements:

- Converting categorical data to numerical data.
- Standardizing and normalizing the data.

Data:

- Data Labels Used: Gender, Education, Experience, Age
- Data Modifications:
  - Gender: Male(0) Female(1)
  - Education: Bachelor's(1) Master's(2) PhD(3)

Before Improvements:

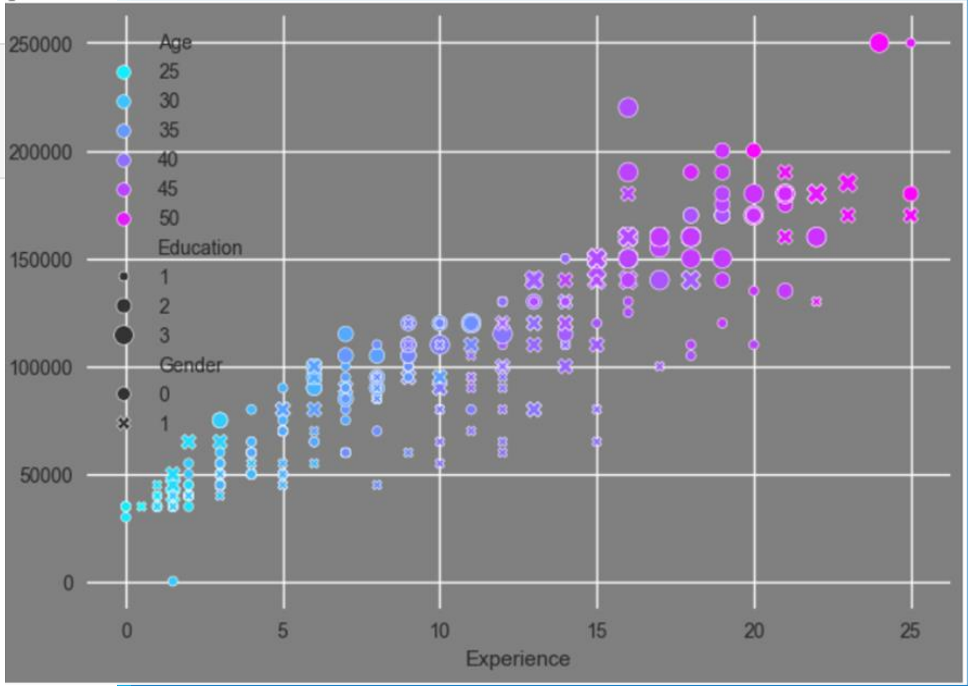
```
acc = round((knn_model.score(X_test, Y_test) * 100), 2)
print(f'Accuracy : {acc} %')
```

```
Accuracy : 42.11 %
```

After Improvements:

```
acc = round((knn_model.score(X_test, Y_test) * 100), 2)
print(f'Accuracy : {acc} %')
```

```
Accuracy : 47.37 %
```



K-NEAREST NEIGHBORS

Model Analysis:

- Knn is not the most fit model for this data set.
- It struggles with data that includes several input variables.

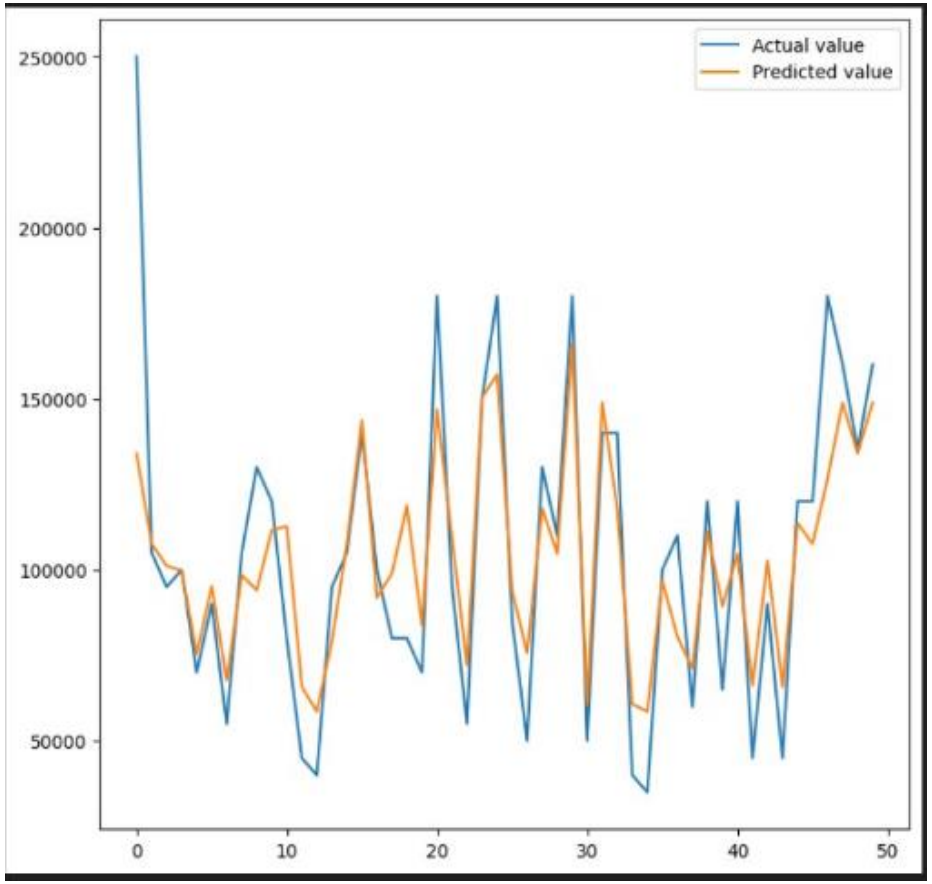
My Experience:

- Overall, the model was pretty simple to implement.
- Challenges: Data prep
- Solution: Work with only numerical data
- I learned that it is important to choose a model that is best fit for the data. Some models may produce a more or less accurate result given your data set.

XGBoost

- Initially, I did Sklearn approach for my model and gained a score of 73%.
- However, my training data score is r^2 is 98%, which indicate overfitting. Training data can't be too far off from the testing data, at least a 5% difference.
- The overfitting is caused in how small the dataset whereas XGBoost is powerful algorithm that mainly works with larger dataset.
- So, I decided taking the API route of XGBOOST:
    - r^2 score of 74%
    - But my training data score in r^2 is 85%
    - What was different? Using a Dmatrix(), a data structure, and implementing n-estimators to 12 trees, given that fewer trees prevent overfitting.

XGBOOST



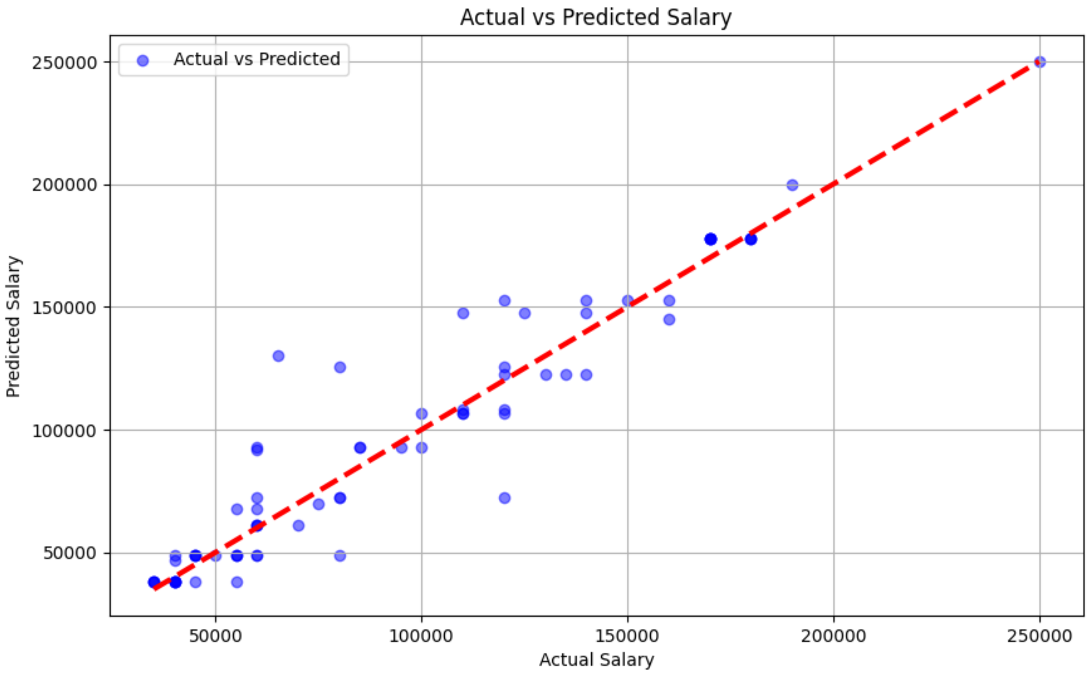
Would you say your algorithm is ideally the best algorithm for this dataset?

XGBoost is very powerful algorithm where you can hold power in how efficient your model can model through hyperparameter tuning, but it was not ideal for this dataset as the algorithm is too complex for a simpler dataset.

Decision Tree

- Used test size of 0.2 (20%) and depth of 5
- Seeded to reproduce random results (seed=100)
  - Model Score = 89.87
  - R^2 = 90.36

```
Model Score :  89.86882126612232
r2 :  90.35652660939402
Actual: 175000.0, Predicted: 175892.85714285713, Difference: 892.8571428571304
Actual: 170000.0, Predicted: 175892.85714285713, Difference: 5892.85714285713
Actual: 100000.0, Predicted: 106734.69387755102, Difference: 6734.693877551021
Actual: 90000.0, Predicted: 106734.69387755102, Difference: 16734.69387755102
Actual: 40000.0, Predicted: 40312.5, Difference: 312.5
Actual: 95000.0, Predicted: 106734.69387755102, Difference: 11734.69387755102
Actual: 65000.0, Predicted: 40312.5, Difference: -24687.5
Actual: 140000.0, Predicted: 153793.10344827586, Difference: 13793.103448275855
Actual: 35000.0, Predicted: 40312.5, Difference: 5312.5
Actual: 150000.0, Predicted: 153793.10344827586, Difference: 3793.103448275855
```



My Experience

- Decision Tree is good for this data set because: it can handle non-linear relationships, use numerical and categorical variables, and isn't heavily affected by outliers
- However, it is prone to overfitting + high variance. It also isn't suited to extrapolate salaries not within the training data

I learned: more Python, cleaning/hot encoding data, pandas, numpy, matplotlib

ANAYSIS

Highest Accuracy?

Random forest

Most optimal: given Accuracy and Program friendliness?

Linear Regression; since it's easier to implement, not overfitting, and r^2 of 84-89%.

OUR MACHINE LEARNING OUTLINE

1. Finding the best algorithm for the dataset.
2. Look at the data.
3. Turn raw data to clean data.
4. Model Implementation.
5. Test your model and visualize the output.

THANK YOU

GitHub: CSS-Exploring-Machine-Learning-Models/Machine_Learning_Algorithms : . (github.com)