Data Science – Machine Learning

A Salary Prediction Model – by Phil Baltazar

Salary Prediction Model
at a glance

The model predicts salaries of jobs based on attributes (features) present in a dataset.
It is summarized in 4 parts:

- Define
- Discover
- Develop
- Deploy

The dataset has 1 million datapoints normally distributed, and each feature is explored further.

Developed and designed by Phil Baltazar. https://github.com/pbswe

Part 1 - Define

the problem

The problem:
“We need to accurately predict the salary of a job based on its attributes (level, experience, industry, location) to pay our employees fairly and remain competitive in the market.”

The solution:
“Develop a Machine Learning algorithm that provides an accurate salary recommendation based on the attributes above, with a high score of efficacy.”

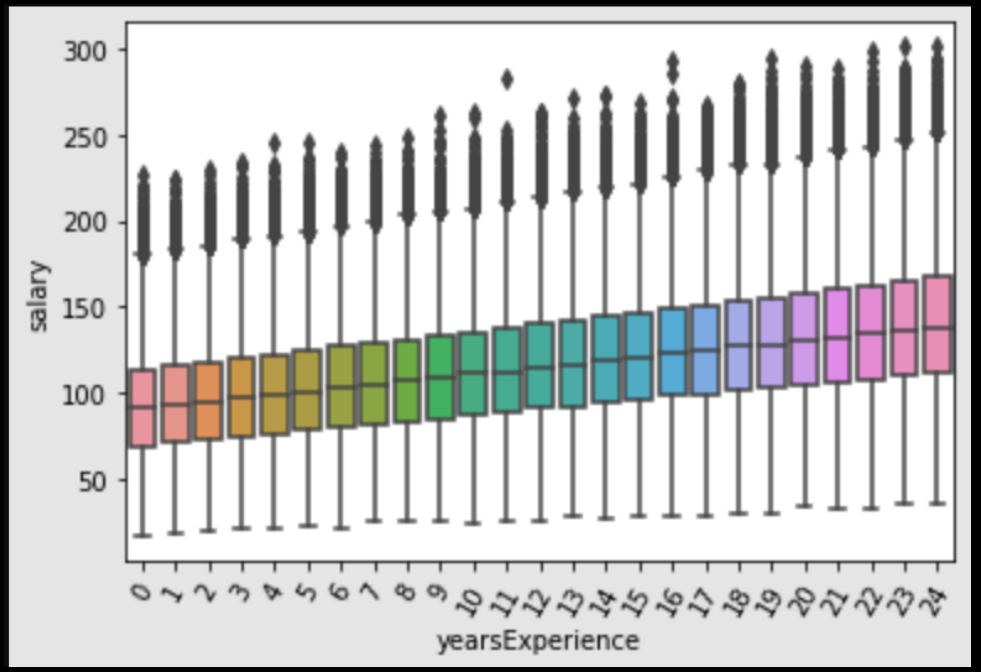
Tools used:
Jupyter Notebook (Anaconda) to write Python scripts, Tableau for additional visualization. Then, several libraries were loaded to assist with calculations, EDA (exploratory data analysis), data framing, visualization plots and machine learning fine tuning.

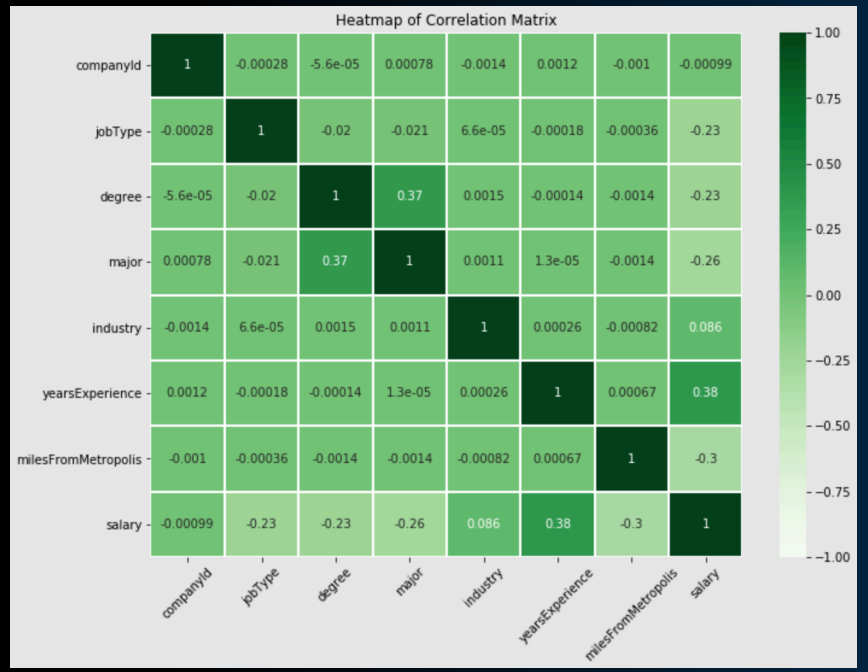
Libraries included:
pandas, numpy, matplotlib, seaborn, scipy, and several from sklearn such as pre-processing, linear regression, decision tree, random forest and gradient boosting.

Part 2 - Discover

the data

After loading the data, the model cleaned and sanitized the datapoints, checking for duplicates, null values and correcting data types. Then, it moved on to discover the strongest correlation between features and salary, plotting data to help visualization.



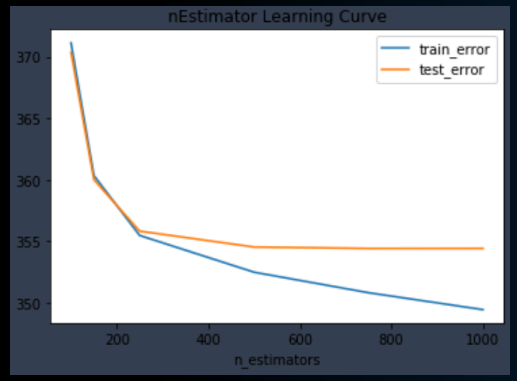


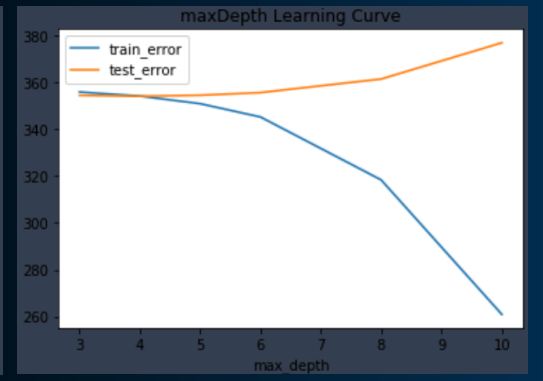
Part 3 - Develop

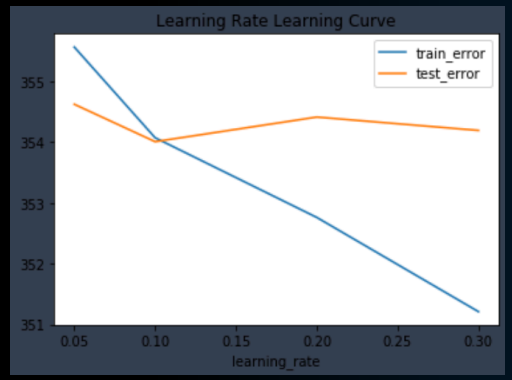
the model

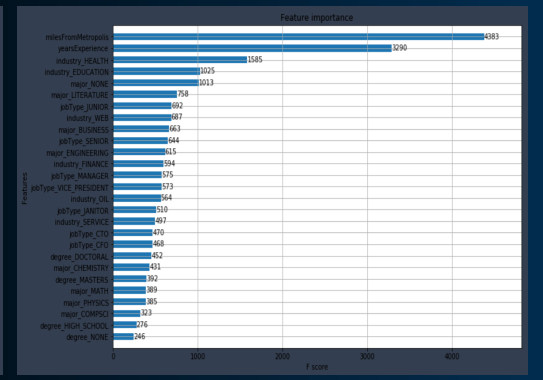
Now that the data is cleaned up and better understood, the model engineers features to feed into the machine learning algorithms. Then, the algorithms were fined tuned to increase the accuracy the model predicts a salary based on the information given.

```
In [178]:
# Looping through different models to obtain their MSE. Hyperparameters chosen manually (testing).

models = []
meanMse = {}

lr = LinearRegression()

sgd = SGDRegressor(max_iter=200, learning_rate='optimal')

dtr = DecisionTreeRegressor(max_depth=15)

rfr = RandomForestRegressor(n_estimators=150, n_jobs=-1, max_depth=30,
                            min_samples_split=60, max_features='sqrt')
xgb = xgboost.XGBRegressor(max_depth=5, n_estimators=500, learning_rate=0.1, n_jobs=-1)

models.extend([lr, sgd, dtr, rfr, xgb])

print('Cross Validation of Models Initiated...\n')

for model in models:
    mseIter = evalModel(model)
    meanMse.update({model:mseIter})

bestModel = min(meanMse, key=meanMse.get)

print('\n\nThe model with the lowest average MSE to use for predictions is:\n')
print(bestModel)

Cross Validation of Models Initiated...
```









Developed and designed by Phil Baltazar. https://github.com/pbswe

Part 4 - Deploy

the solution

The best ML model for this project was XGBoost (Extreme Gradient Boosting). After fine tuning these algorithms, the model begins to deploy the solution for further processing.

In [204]:

```
# Creating a function that trains model on entire training set and saves the model to disk.

def trainedCleanModelDF(raw_train_features, raw_train_targets):

    # Loading csv files
    featuresDF = pd.read_csv(raw_train_features)
    targetsDF = pd.read_csv(raw_train_targets)

    # Cleaning feature and target dataframes per analysis above
    df = pd.merge(featuresDF, targetsDF, on='jobId')
    df = df[df['salary'] > 8.5]
    categoriesDF = df[['jobType', 'degree', 'major', 'industry']]
    categoriesDF = pd.get_dummies(categoriesDF, drop_first=True)
    featuresDF = pd.concat([categoriesDF, df[['yearsExperience', 'milesFromMetropolis']]], axis=1)
    targetsDF = df[['salary']]
    del categoriesDF, df

    # Implement best model discovered per analysis above
    model = xgboost.XGBRegressor(learning_rate=0.05, max_depth=4, n_estimators=1500)
    model.fit(featuresDF, targetsDF)

    # Save model to disk
    model_file = 'model'
```

In [205]:

```
# Script that prepares data, predicts salaries, and exports results
# This class will be saved in a .py file for private use - See "Salary_Prediction_Module.py"
class salaryPredictionModel():

    # Read the 'model' file which was saved
    def __init__(self, model_file):
        self.xgb = pickle.load(open(model_file, 'rb'))

    # Takes data, prepares data, makes predictions from trained model, and exports results to csv f
    def exportPredictions(self, data_file):

        # Load csv file
        df_pred_features = pd.read_csv(data_file)

        # Saves jobId column for output file
        df_pred_jobId = pd.DataFrame(df_pred_features['jobId'])

        # Prepares data to be fed into the model
        df_pred_categories = df_pred_features[['jobType', 'degree', 'major', 'industry']]
        df_pred_categories = pd.get_dummies(df_pred_categories, drop_first=True)
        df_pred_features = pd.concat([df_pred_categories, df_pred_features[['yearsExperience','mile
        del df_pred_categories

        # Loads model from disk, predicts salaries, and exports results to .csv file
```

Salary Prediction Model

conclusion

The model taught us there are features with stronger correlation to salary than others. YoE and job type (level) are the strongest predictors of salaries, as shown on the Tableau graph: