# Machine Learning Engineer Nanodegree

## Capstone Project

Manish Kumar
October 20th, 2019

# I. Definition

### Project Overview

This project is related to Movie Industry. Movie industry comprises of many uniques features in itself. To work with a dataset i.e. related to this industry is always going to be challenging, intriguing and involving a lot of fun. There are many interesting aspects in it and their influences on whole industry is pretty much evident. And this what we are going to do in this project and it is always more convincing to see the results through data rather than intuitively. There are many research going on in this domain and many research papers already published. A paper published by Simonoff & Sparrow discusses various predictor variables that can affect the gross revenue of a movie e.g. Genre of the film, MPAA(Motion Picture Association of America) rating, origin country of the movie, star cast of the movie, production budget of the movie etc *[Ref.1]* .

The project dataset under consideration is part of *Kaggle Competition Problem* (https://www.kaggle.com/c/tmdb-box-office-prediction) (https://www.kaggle.com/c/tmdb-box-office-prediction)). The dataset contains over 7000 movies, collected from The movie Database(TMDB). The task is to predict these movies overall box office revenue.

The dataset has columns like *budget, genres, overview, popularity, release date, runtime, keywords, cast crew, revenue etc*. Out of 7398 movies, 3000 movies are given as training dataset and remaining 4398 movies as test dataset.

### Problem Statement

In this project the task is to predict target variable (*Revenue*) using independent variables like *budget*, *genres*, *popularity*, *runtime*, *release date*, *cast*, *crew* etc. This problem comes under supervised learning as we have labeled data. Further, This Supervised learning problem is about predicting the target (variable) so I need to build a regression analysis model. I need to train the model on the training data, then testing the model performance using test data and finally predicting the revenue for *unseen* data. In Scikit-learn(The package I chosen to use) package provides many regression algorithms. By taking into account the nature of data I decided to use **SGDregressor** algorithm for this problem. The Model will be used to predict gross *revenue* of unseen data (4398 movies). The final result will contain *movie*id_ and *revenue* in csv file.

### Metrics

I have chosen R-squared score as the evaluation metrics of this project (**a regression analysis**). R-squared is a statistical measure of how close the data are to the fitted regression line. It is also known as the coefficient of determination. R-squared value always lies between 0 and 1 *[Ref.2]*.

The most general definition of the coefficient of determination is

$$(1 - \frac{SS_{res}}{SS_{tot}})$$

, where

$$SS_{tot} = \sum_{i=0}^{n} (y_i - y^{-})^2$$

is total sum of squares(proportional to the variance of the data) and

$$SS_{res} = \sum_{i=0}^{n} (y_i - f_i)^2$$

is the sum of squares of residuals *[Ref.3]*. A model with R-squared value of 0 means that the model is behaving like a naive predictor model(predict average value all the time) and definitely not a good one. However, a value of 1 indicated the model is predicting the target variable accurately. Keeping this criteria in mind my aim in this project to build a regression model with a R-squared score close to 1.

# II. Analysis

## Data Exploration

The training dataset contains 3000 rows and 23 columns and testing dataset contains 4398 rows and 22 columns. The datasets are a good blend of numerical and categorical columns. Some of the important categorical columns are *genres, original*language, production_companies, production_countries, release_date etc_. Similarly, some of the important numerical columns are *budget, popularity, runtime, revenue etc*. A snapshot of the training dataset looks like below:

| id | belongs_to_collection | budget | genres | homepage | imdb_id | original_language | original_title | overview | popularity | ... | release_date | runtime | spoken_languages | status | tagline | title | Keywords | cast | crew | revenue |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | [{'id': 313576, 'name': 'Hot Tub Time Machine ... | 14000000 | [{'id': 35, 'name': 'Comedy'}] | NaN | tt2637294 | en | Hot Tub Time Machine 2 | When Lou, who has become the "father of the In... | 6.575 | ... | 2/20/15 | 93.000 | [{'iso_639_1': 'en', 'name': 'English'}] | Released | The Laws of Space and Time are About to be | Hot Tub Time Machine 2 | [{'id': 4379, 'name': 'time travel'}, {'id': 9... | [{'cast_id': 4, 'character': 'Lou', 'credit_id... | [{'credit_id': '59ac067c92514107af02c8c8', 'de... | 12314651 |

As we know that machine learning algorithms are susceptible to the input data. Hence, it becomes matter of utmost important to wrangle the data before feeding it into the model.

In data exploration step it is important to look for **missing values, duplicate values , null values and outliers** in the given data. Using **describe()** function of Pandas I can easily see various statistics of all numerical features present in the dataset.

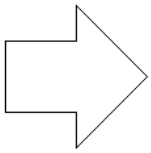
| | budget | popularity | runtime | revenue |
|---|---|---|---|---|
| count | 3000.000 | 3000.000 | 2998.000 | 3000.000 |
| mean | 22531334.110 | 8.463 | 107.857 | 66725851.889 |
| std | 37026086.412 | 12.104 | 22.086 | 137532326.336 |
| min | 0.000 | 0.000 | 0.000 | 1.000 |
| 25% | 0.000 | 4.018 | 94.000 | 2379808.250 |
| 50% | 8000000.000 | 7.375 | 104.000 | 16807068.000 |
| 75% | 29000000.000 | 10.891 | 118.000 | 68919203.500 |
| max | 380000000.000 | 294.337 | 338.000 | 1519557910.000 |

The above chart shows statistics of 3 measure features and 1 target variable. We can see many abnormalities in the data. For variables *budget, popularity, runtime and revenue* the minimum value is 0 and for revenue its 1. If we see maximum value all the variables have abnormally high values. This also signifies towards presence of outliers in the data. However, we can not say those records as outliers without further investigation because the data comes from movies industry which has peculiar features in itself. For example some movies perform exceptionally well in term of gross revenue collection, similarly some movies are made with huge budget etc.

Null values are something that can break the learning model. For 'runtime' variable I found only two such records and I decided to remove those rows.

Floating point values also create problem for regression model so I converted variables like *budget, popularity, runtime and revenue* to integer. In original dataset columns is present in form of list of dictionary, so I need to convert this categorical variable in form of comma separated text so that later stage I can easily convert it into one-hot encoding.

```
0                       [{'id': 35, 'name': 'Comedy'}]
1      [{'id': 35, 'name': 'Comedy'}, {'id': 18, 'nam...
2                        [{'id': 18, 'name': 'Drama'}]
3      [{'id': 53, 'name': 'Thriller'}, {'id': 18, 'n...
4      [{'id': 28, 'name': 'Action'}, {'id': 53, 'nam...
5      [{'id': 16, 'name': 'Animation'}, {'id': 12, '...
6      [{'id': 27, 'name': 'Horror'}, {'id': 53, 'nam...
7                  [{'id': 99, 'name': 'Documentary'}]
8      [{'id': 28, 'name': 'Action'}, {'id': 35, 'nam...
9      [{'id': 35, 'name': 'Comedy'}, {'id': 10402, '...
10                      [{'id': 18, 'name': 'Drama'}]
```



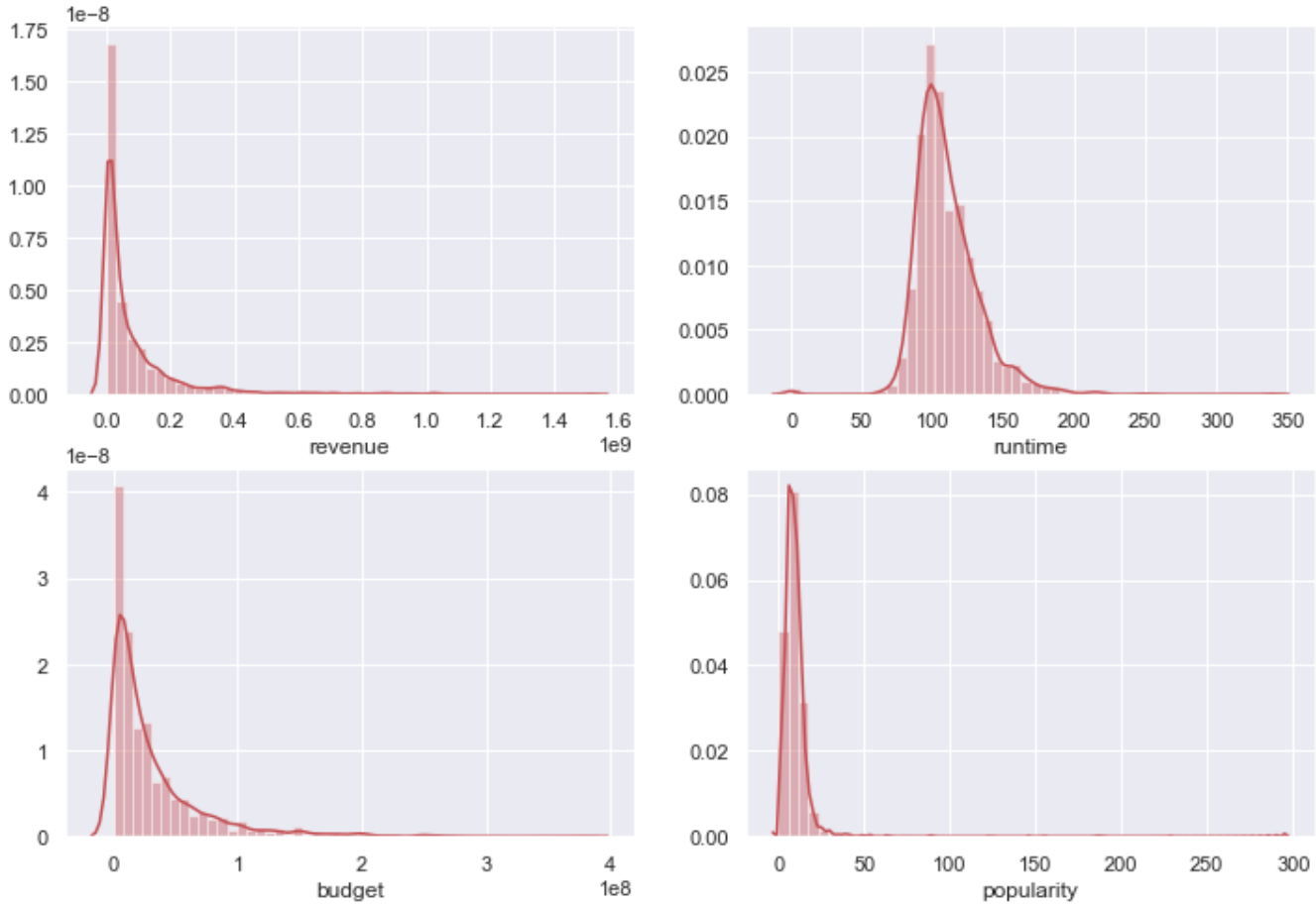
```
0                                                Comedy
1                                Comedy,Drama,Family,Romance
2                                                 Drama
3                                         Thriller,Drama
4                                        Action,Thriller
5                                Animation,Adventure,Family
6                                        Horror,Thriller
7                                           Documentary
8                    Action,Comedy,Music,Family,Adventure
9                                         Comedy,Music
10                                                Drama
```

In the original dataset there is one columns 'release_date' which denotes the releasing date of the movie. This is in form of date data type. This columns is not of much of use for my model in this form, However new features can be created using this column like 'release_month', 'day_of_week' etc.

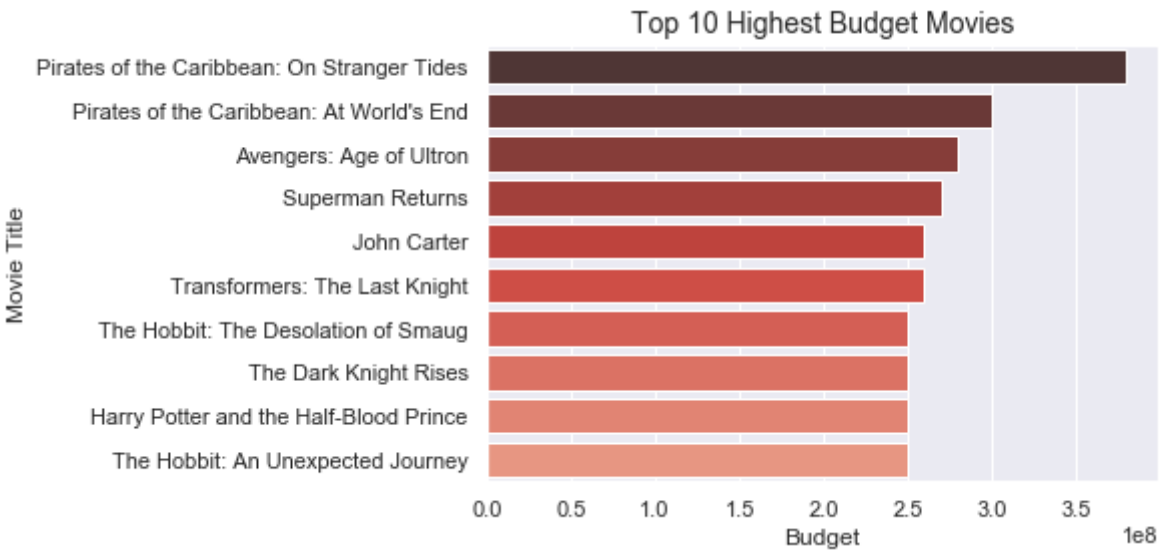
One of the main aspects of the data data exploration is to find out important features for the current problem from the available data. This can be done using univariate, bivariate and multivariate analysis of the data. These analysis I done using **Pandas, Seaborn and Matplotlib** libraries. The data visualization analysis is discussed in the next section.

## Exploratory Visualization

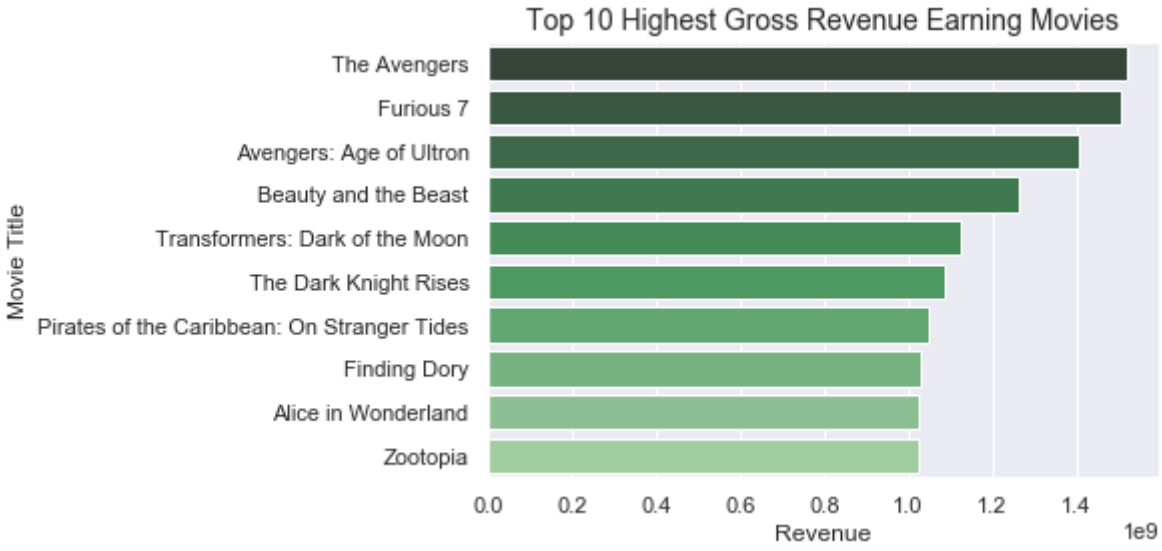
In data Exploratory analysis, first of all I will discuss univariate analysis. My focus is on mainly 4 variables i.e *budget*, *runtime*, *popularity* and *revenue*. The below histogram shows distribution curve of these variables.



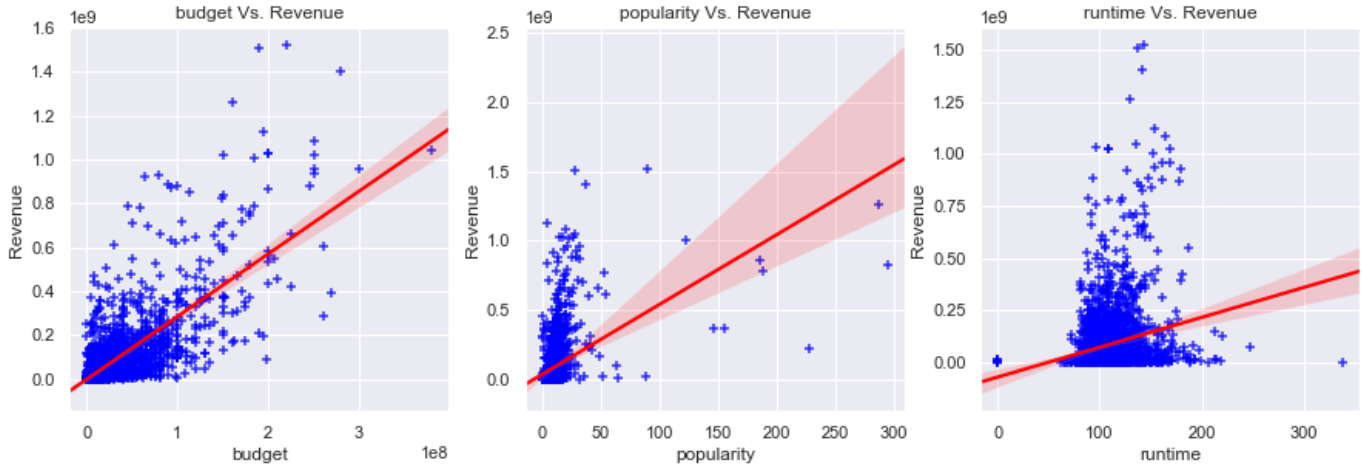
It is evident from the graph that curve of variables *budget*, *revenue*, and *popularity* are right-skewed distribution and distribution graph of *runtime* is a normal distribution. The right-skewed graph signifies a possibility of presence of outliers in case of *budget*, *revenue*, and *popularity*. A further analysis of the above variables shows that these are genuine records, so I have to leave them as it is. The below bar chart shows budget of top ten movies respectively.




Also, The below bar chart shows revenue of top ten movies respectively.

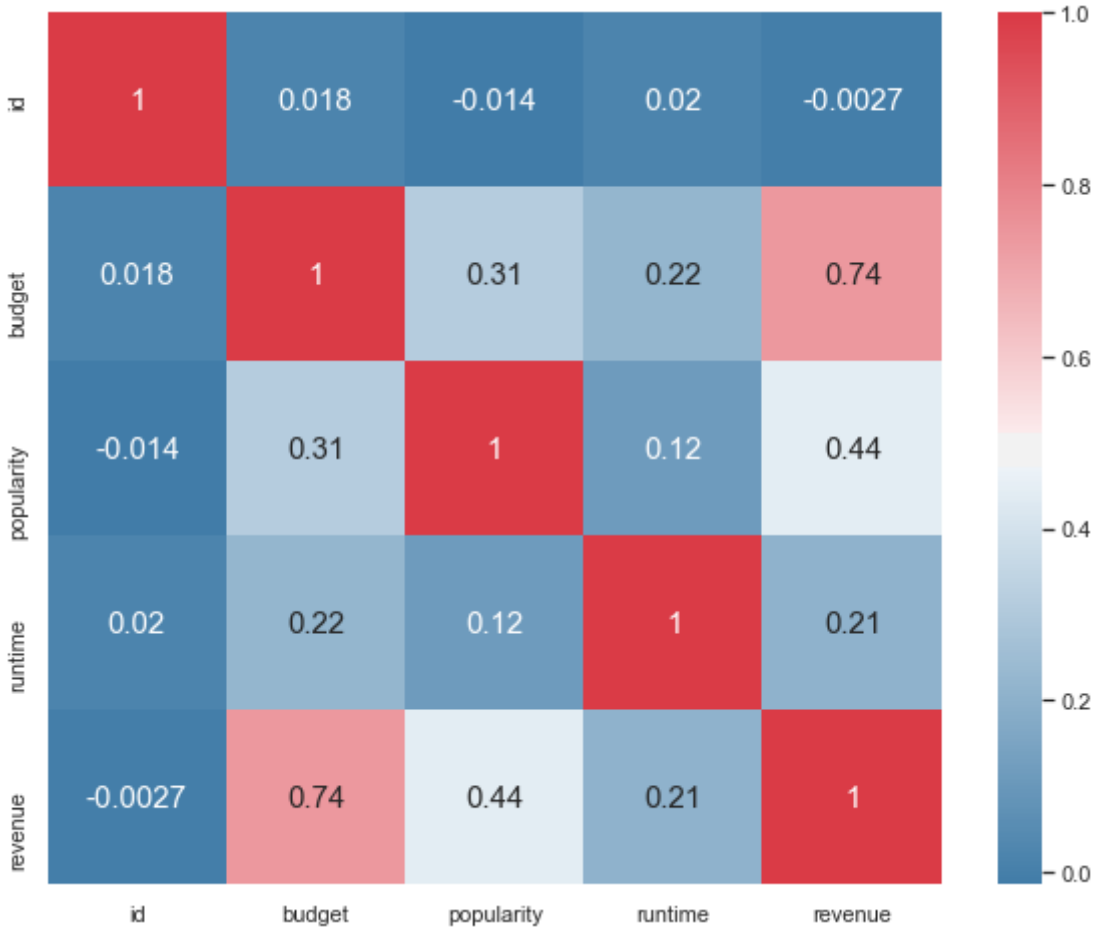



In bivariate analysis, my intention is to see the correlation between the features(*budget*, *runtime*, *popularity*) and the target variable(*revenue*). The below graph shows the scatter-plots of the these important features against *revenue*.



From the above plots signifies a strong correlation between *budget* and *revenue*. There is also correlation between *popularity* and *revenue*, and *runtime* and *revenue* to some lesser extent.

We know that in regression model correlation between features and target variable is an important aspects to look upon while dong the EDA. Therefore, By doing a multivariate analysis using heatmap we can see the correlation between different variables at one place in summarized form. The below plots shows heatmap of the numerical variables of the dataset.




As discussed earlier, in above heatmap it is evident a strong correlation between *budget* and *revenue*. There is also correlation between *popularity* and *revenue*, and *runtime* and *revenue*.

## Algorithms and Techniques

In this project I used mainly two regression algorithm i.e. linear regression and stochastic gradient descent regression algorithm. The simple linear regression algorithm establish linear relation between features and target variable. The simple linear regression between feature *x_ and target variable _y* is modeled as below equation:

$$y_i = b_0 + b_1 x_1$$

Here, b_0 is called intercept i.e. expected value of target variable when the value of feature *x_ is 0. b*1 is called slope i.e. the expected change in the target variable for each 1 unit increase in the feature variable *x*. Similarly, when we try to predict value of target variable using multiple features then then it is called multiple linear regression. This can be modeled in form of equation as below:

$$y_i = b_0 + b_1 x_1 + b_2 x_2 + b_3 x_3 + \ldots$$

. The fitting of a regression line is done by minimizing the sum of the squared difference between true value and predicted value,

$$(y - y_i)$$

which is called residual. The error or cost function is given as below:

$$SS_{res} = \frac{1}{2}\sum_{i=0}^{n}(y_i - f_i)^2$$

Different techniques can be used to minimize the cost function. One of them optimization algorithm is gradient descent.

We can use gradient descent to find the values of the model's parameters that minimize the value of the cost function. Gradient descent iteratively updates the value of the model's parameters by calculating the partial derivatives of the cost function at each step *[Ref.4]*. Local minima problem can occur in case of gradient descent if the surface if non-convex. However, it works fine for the above cost function as it is convex in nature. Learning rate is an important hyper-parameter of gradient descent. The gradient descent steps consist of subtracting the gradient of the error times the learning rate &aplha;. If the learning rate is small enough, the cost function will decrease with each iteration until gradient descent has converged on optimal parameters. If the learning rate is too large, the gradient descent could oscillate around the optimal values of parameters *[Ref.4]*. There are two varieties of gradient descent: Batch gradient descent and Stochastic gradient descent.
In this project I am using SGDRegressor of scikit-learn as the proposed model.

### Benchmark

For benchmark model, I used one of the linear regression algorithm. This benchmark model will work as beseline for the problem.

```
from sklearn.linear_model import LinearRegression
reg_b=LinearRegression()
reg_b.fit(X_train,y_train)
y_pred_b=reg_b.predict(X_test)
```

The r2_score for the above *Benchmark Model* is : **0.63787**

```
print("Benchmark Model Score: {0:.5f}".format(model_eval(y_test,y_pred_b)))
```

# III. Methodology

### Data Preprocessing

Data preprocessing is an important step in machine learning model building process. As I mentioned in the data exploration step that machine learning algorithms are very sensitive towards the input data. Hence, it becomes essential to pre-process the data for the better model performance.

Also, It is important to note that categorical data in form of text breaks the machine learning algorithm, so the categorical data first needs to be converted into numerical data using one-hot encoding method. In Pandas software package there is a function get_dummies() using which I can create one-hot encoding very easily.

```
# Creating one-hot encoding for categorical variable 'release_month' and 'day_of_w
eek'
import pandas as pd
features_1=pd.get_dummies(features)

# Creating one-hot encoding for the variable 'genres'
import pandas as pd
tmdb_train_df=pd.concat([tmdb_train_df,tmdb_train_df.genres.str.get_dummies(sep
=',')],axis=1)
```

after doing one-hot encoding the dataset will look like this:

| | budget | popularity | runtime | Action | Adventure | Animation | Comedy | Crime | Documentary | Drama | ... | release_month_Nov | release_month_Oct | release_m |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 14000000 | 6 | 93 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | ... | 0 | 0 | |
| 1 | 40000000 | 8 | 113 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | ... | 0 | 0 | |
| 2 | 3300000 | 64 | 105 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | ... | 0 | 1 | |
| 3 | 1200000 | 3 | 122 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | ... | 0 | 0 | |
| 4 | 0 | 1 | 118 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | |

In this project my proposed model is SGDregressor from the scikit-learn package.

Stochastic gradient descent is sensitive to feature scaling, so it is highly recommended to scale the data. In this case, it is important to standardize the data(mean=o and variance=1). Also, the scaling to be done for training and testing features both to get the meaningful results[Ref.2]. Here I have used StandardScaler() function from the sklearn.preprocessing module to standardize the data.

```
# standardizing the testing and training features
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
scaler.fit(X_train)
X_train=scaler.transform(X_train)
X_test=scaler.transform(X_test)
```

## Implementation

For this project to solve the underlying problem I built a regression model. The software package used is Scikit-learn as it provides a vast varieties of machine learning algorithms. For benchmark model, I used linear regression algorithm. This benchmark model will work as beseline for the problem.

**Benchmark Model:**

```
from sklearn.linear_model import LinearRegression
reg_b=LinearRegression()
reg_b.fit(X_train,y_train)
y_pred_b=reg_b.predict(X_test)
```

Here, I called *LinearRegression()* of sklearn.linear_model. This model is based on Ordinary Least Squares algorithm. *LinearRegression()* returns linear regression object **reg_b** using which we can fit the model.

**Proposed Model:**

As for the proposed solution for this project I chosen SGDregressor (Optimization of Linear Regression) algorithm seeing the nature of the dataset.

```
from sklearn import linear_model
reg_p=linear_model.SGDRegressor()
reg_p.fit(X_train,y_train)
```

The hyper-parameters of the model are as follows:

```
SGDRegressor(alpha=0.0001, average=False, epsilon=0.1, eta0=0.01,
       fit_intercept=True, l1_ratio=0.15, learning_rate='invscaling',
       loss='squared_loss', n_iter=5, penalty='l2', power_t=0.25,
       random_state=None, shuffle=True, verbose=0, warm_start=False)
```

Finally, after fitting of the model the regression object (**reg_p**) is used to predict the target variable values.

```
y_pred_p=reg_p.predict(X_test)
```

## Refinement

I got output of the benchmark model and proposed model as shown below:



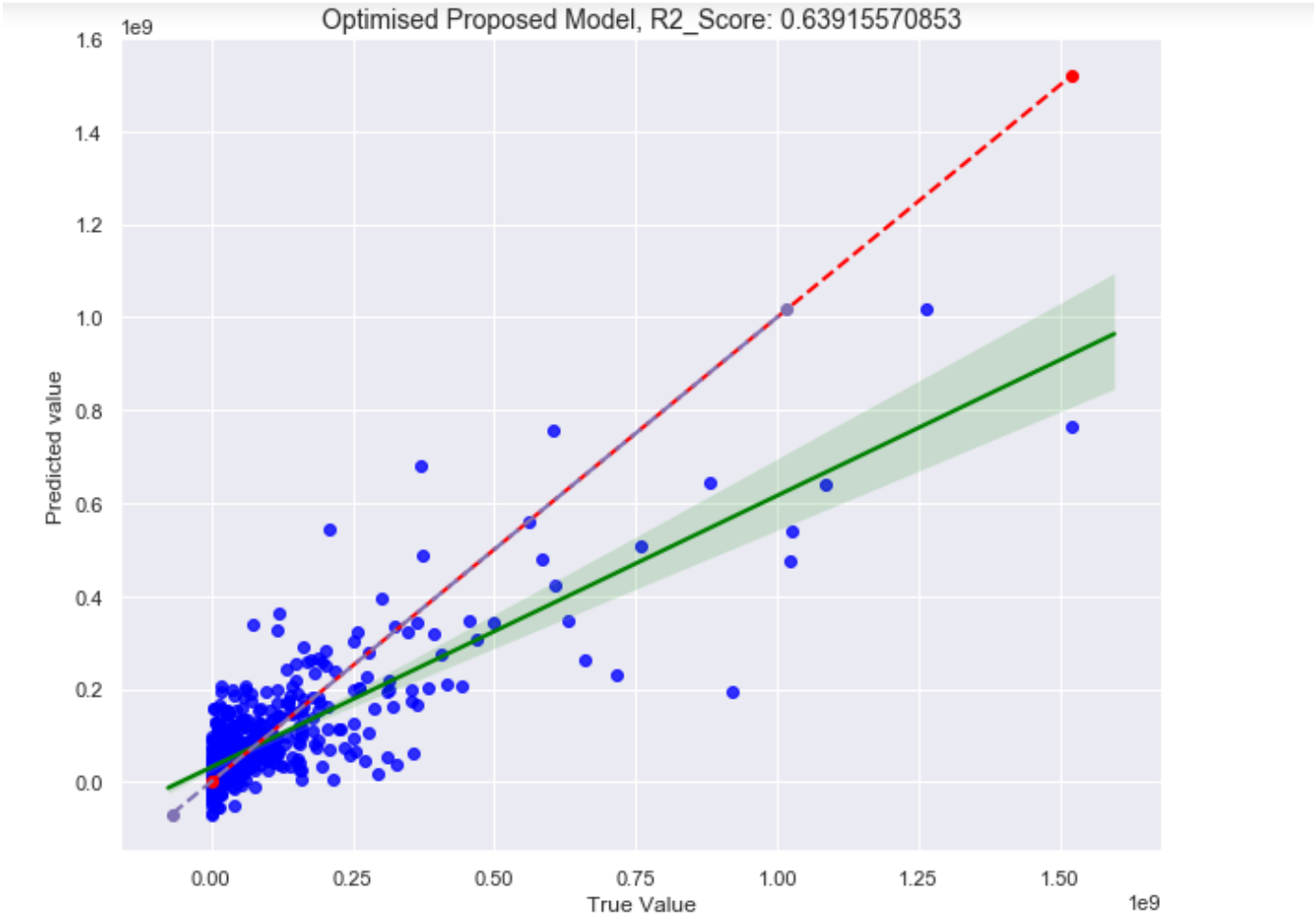
To further improve the performance of the model hyper hyper-parameters tuning is required. Hyper-parameter tuning is one of the important aspects of model building process. An important hyper-parameter of gradient descent is the learning rate. Other important hyper-parameter in SGDregressor are *loss function, penalty, epsilon, n*iter, alpha etc_. These hyperparmaters need to be tuned to get the best performance that built in previous section. Manually finding best combination of hyper-parameter is a cumbersome task. Therefore, I will use **GridSearchCV()** to find out the best set of hyper-parameters.

```
parameters={'loss':['squared_loss','huber'],
            'penalty': ['l2'],
            'learning_rate':['optimal','invscaling'],
            'n_iter':[5],
            "alpha": [0.0001, 0.001, 0.01, 0.1, 1, 10, 100]
            }
grid = GridSearchCV(reg_p,parameters)
grid_reg = grid.fit(X_train,y_train)
best_reg = grid_reg.best_estimator_
```

Using the above method I got the below result:

```
SGDRegressor(alpha=0.0001, average=False, epsilon=0.1, eta0=0.01,
       fit_intercept=True, l1_ratio=0.15, learning_rate='invscaling',
       loss='squared_loss', n_iter=5, penalty='l2', power_t=0.25,
       random_state=None, shuffle=True, verbose=0, warm_start=False)
```

Using the above regression object I built the refined proposed model and got slightly better result than the previous proposed model.




# IV. Results

## Model Evaluation and Validation

As mentioned in the previous section, I chose R-squared score as the evaluation metric for the model that I developed in the previous step. In sklearn, we have libraries using which r2_score can be calculated very easily.

```
from sklearn.metrics import r2_score
score=r2_score(y_true,y_predict)
```

By comparing this score value I can evaluate the model and further hyper parameter tuning can be done to improve the r2_score of the model. The final model is developed after the hyper-parameter as shown below:

```
best_reg=SGDRegressor(alpha=0.1, average=False, epsilon=0.1, eta0=10,
       fit_intercept=True, l1_ratio=0.15, learning_rate='optimal',
       loss='squared_loss', n_iter=5, penalty='l2', power_t=0.25,
       random_state=None, shuffle=True, verbose=0, warm_start=False)
y_pred_op=best_reg.predict(X_test)
```

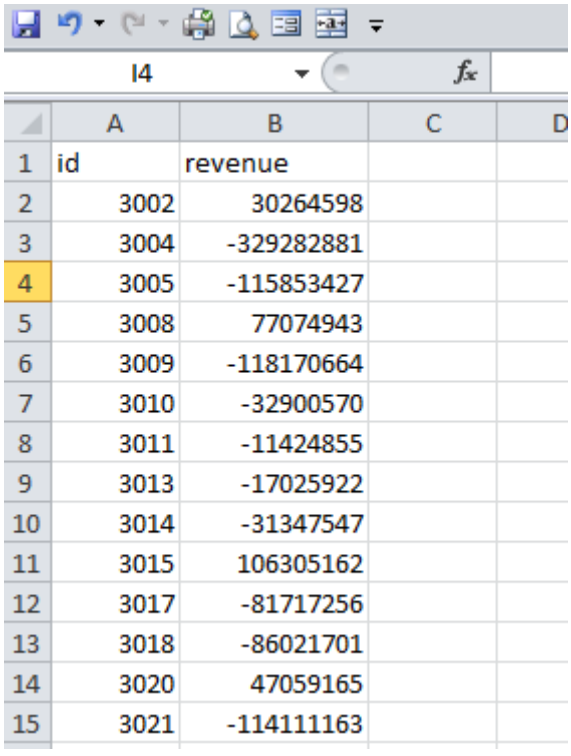
The final output looks like as shown below:


I got the the **r2_score** of the *Benchmark model, proposed model and refined proposed model as below:*

```
Benchmark Model Score: 0.63787
Proposed Model Score: 0.63308
__Optimized Proposed Model Score: 0.63916__
```

Here, on the underlying data LinearRegression model(i.e. *Benchmark* ) performance is quite good. I was able to get only a slight improvement over benchmark model using the proposed model (*SGDRegressor*).

The proposed model performance on **unseen** data is done and results are as shown below:



I4

| | A | B | C | D |
|---|---|---|---|---|
| 1 | id | revenue | | |
| 2 | 3002 | 30264598 | | |
| 3 | 3004 | -329282881 | | |
| 4 | 3005 | -115853427 | | |
| 5 | 3008 | 77074943 | | |
| 6 | 3009 | -118170664 | | |
| 7 | 3010 | -32900570 | | |
| 8 | 3011 | -11424855 | | |
| 9 | 3013 | -17025922 | | |
| 10 | 3014 | -31347547 | | |
| 11 | 3015 | 106305162 | | |
| 12 | 3017 | -81717256 | | |
| 13 | 3018 | -86021701 | | |
| 14 | 3020 | 47059165 | | |
| 15 | 3021 | -114111163 | | |

## Justification

If I compare the evaluation metrics of the the models developed in this project.

```
Benchmark Model Score: 0.63787
Proposed Model Score: 0.63308
__Optimized Proposed Model Score: 0.63916__
```

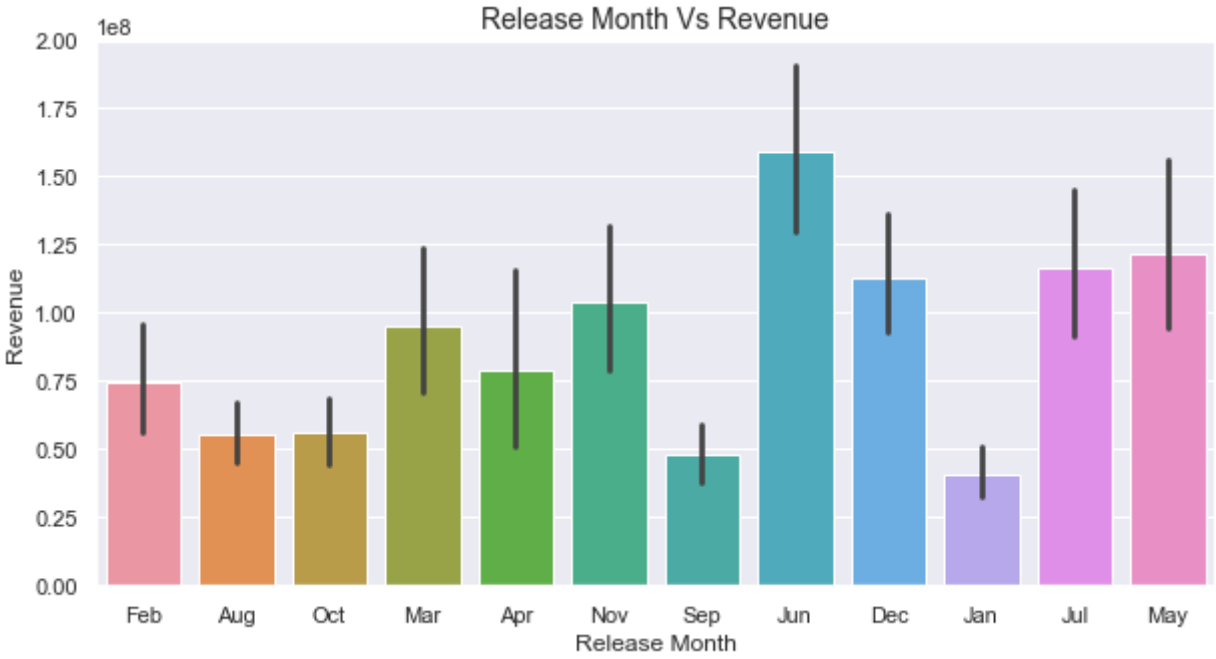
I see that the **proposed model** performing as good as the *Benchmark Model* in most cases and even beating it by a slight margin.

# V. Conclusion

## Free-Form Visualization

As already discussed, the given dataset in this project has a lot peculiar features. Most of them are are already mentioned the **Exploratory Visualization** section.

The bar chart shown below shows some of the interesting facts about the movie releasing month and the *revenue*.




Movies released in the *June* month shows highest revenue and *January* shows lowest. May be because people go out more in *Summer that the chilling Winter!*

## Reflection

In this project, task was to predict the overall gross *revenue* of the movies using the historical data. Two different set of datasets are provided namely *train.csv* and *test.csv*.

The training dataset has some features that have high correlation with the *revenue*. The dataset also provides opportunity of many feature extraction from its given columns. Overall the dataset offers a lot of challenges in terms of wrangling and feature extraction.

One more challenge I faced in this project in fine tuning the hyper-parameter of the proposed model *SGDRegressor*.

## Improvement

There is certainly scope of improvement in the present *proposed solution* of this project.

In the proposed model I used *SGDRegressor* that performs better if more training data points are given to the model and also, if we can improve the number of *features*.

There are regression algorithms in the scikit-learn that needs to be explored e.g *Lasso, ElasticNet, RidgeRegression, etc.* to find out a better model than the proposed solution.

## References:


[1] Simonoff, Jeffrey S. and Sparrow, Ilana R., Predicting movie grosses: Winners and losers, blockbusters and sleepers, 1999 (https://archive.nyu.edu/handle/2451/14752 (https://archive.nyu.edu/handle/2451/14752))

[2] Regression Analysis: How Do I Interpret R-squared and Assess the Goodness-of-Fit?, 30 may 2013, (https://blog.minitab.com/blog/adventures-in-statistics-2/regression-analysis-how-do-i-interpret-r-squared-and-assess-the-goodness-of-fit (https://blog.minitab.com/blog/adventures-in-statistics-2/regression-analysis-how-do-i-interpret-r-squared-and-assess-the-goodness-of-fit))

[3] https://en.wikipedia.org/wiki/Coefficient_of_determination (https://en.wikipedia.org/wiki/Coefficient_of_determination)

[4] Mastering Machine Learning with scikit-learn, 2014, Gavin Hackeling.

[5] Stochastic Gradient Descent, (https://scikit-learn.org/stable/modules/sgd.html (https://scikit-learn.org/stable/modules/sgd.html))

[6] TMDB Box Office Prediction, Can you predict a movie's worldwide box office revenue? (https://www.kaggle.com/c/tmdb-box-office-prediction (https://www.kaggle.com/c/tmdb-box-office-prediction))