Machine learning Phase 1

Team id: CS_3

Team Members:

Student Name	**Seat Number**	**Department**
مينا خليفة جندي خله	20201700892	Computer Science
مينا لطفي فايز عبد الله	20201700895	Computer Science
كيرلس عزيز جلال عزيز	20201701116	Computer Science
ماري سعد يوسف سعد سليمان	20201700631	Computer Science
مينا انيس شكري معوض	20201700889	Computer Science
ميشيل مجدي حلمي زرق	20201700886	Computer Science

1. Preprocessing

Data Gathering:

- The data has been gathered by reading two different CSV files using the Pandas library of Python. The first file, "movies-regression-dataset.csv," The second file, "movies-credit-students-train.csv," These two CSV files have been merged based on the ID column to form a single dataset. (We changed the column name of IDs in the second CSV which named (movie_id to id) to facilitate the merging process)

❖ Upon reviewing the data, we conducted a thorough analysis of the null and unique values present in each column. Subsequently, we implemented the following steps to refine the dataset:

1. **IDs column**: We opted to drop this column as it contains unique values that are only beneficial for merging purposes.
2. **Duplicate Title Column**: Given that there were two title columns from merging, we removed one of them, as they are duplicates.
3. **Homepage column**: Due to the presence of numerous null values, we eliminated this column, as it did not provide any useful data.
4. **Tagline Column**: Similar to the Homepage column, we removed the Tagline column due to the significant number of null values. Furthermore, the remaining data consisted of unique values, which did not yield any useful insights.
5. **Runtime Column**: We observed that only one null value was present in the Runtime column. Consequently, we decided to drop it from the dataset.
6. **Status Column**: With only two unique values present, and one of them appearing only once, we deemed this column to be of little use in our analysis.
7. **Release Date Column**: In an effort to further analyze the dataset, we partitioned the Release Date column into three separate columns: Days, Month, and Year.

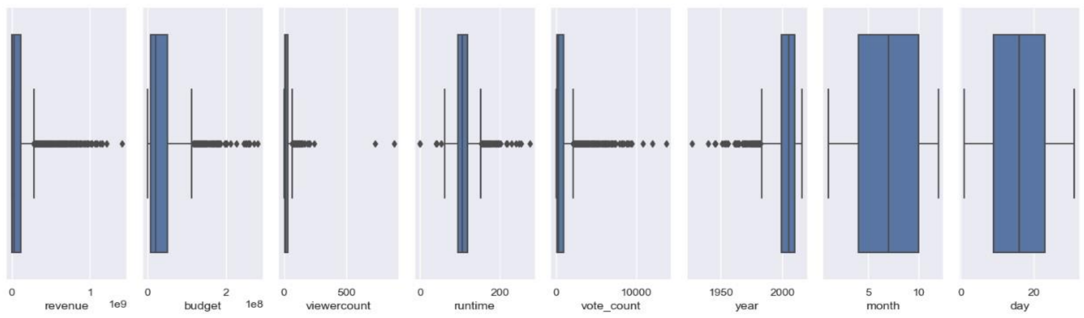
Data Split:

- The next step is to split the data into training and testing sets. The dataset has been split into 80% training data and 20% testing data using the train_test_split function of Scikit-Learn library. The training set has 2425 rows and 15 columns, while the testing set has 607 rows and 15 columns.

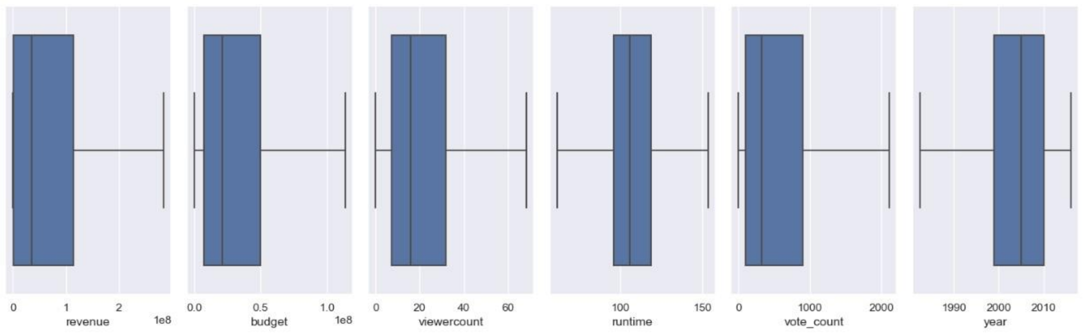
Handling Outliers:

- We conducted an exploratory data analysis by plotting the data points in the X_train and X_test columns with numerical values. Our analysis revealed the presence of outliers in six columns: "**revenue**," "**budget**", "**viewer count**", "**runtime**," "**vote count**", "**year**", "**month**" and "**day**" columns. To address this, we utilized a function called "replace_outliers," which uses the interquartile range method to replace upper and lower outliers with the respective upper and lower limits. These limits were saved for future use in removing outliers from the X_test set.
- The replace_test_outlier function has been created to replace the outliers in the testing data. The function takes the name of the column and the list of limits (we applied the limits obtained from X_train to remove the outliers in the X_test set.) as input and replaces the outliers in the testing data with the limits.

- **Before Handling Outliers**



- **After Handling Outliers**

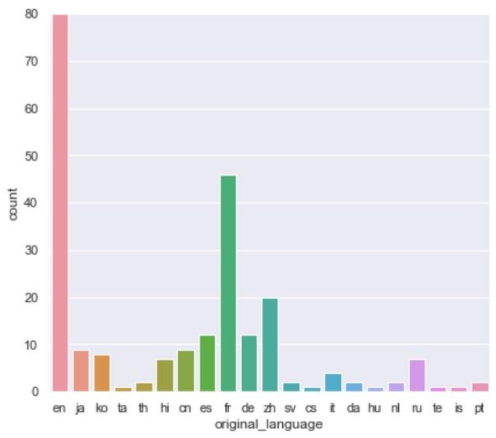


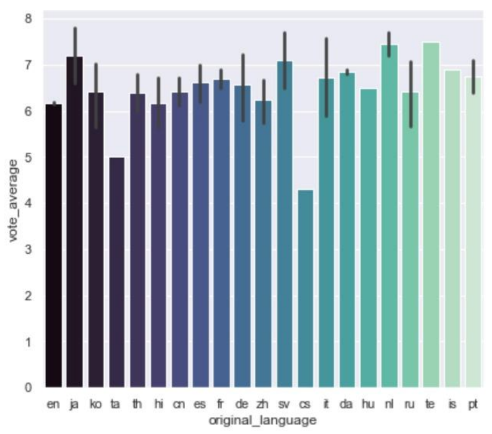
Handling Missing:

- To address the issue of missing values, it was observed that certain columns such as budget and runtime had 0 values, which did not make sense. Therefore, the missing values in these columns were replaced with the mean value of their respective columns in X_train. The mean value of each column was saved to replace the missing values in X_test.

Preprocessing of the original Language column using Ordinal encoder:

- The Language column contained a limited number of unique values (in X_train in total). As a result, an ordinal encoder was applied to transform this column. The fit transform method was used on X_train and transform method on X_test. If any new category was found in X_test, it was replaced with -1.





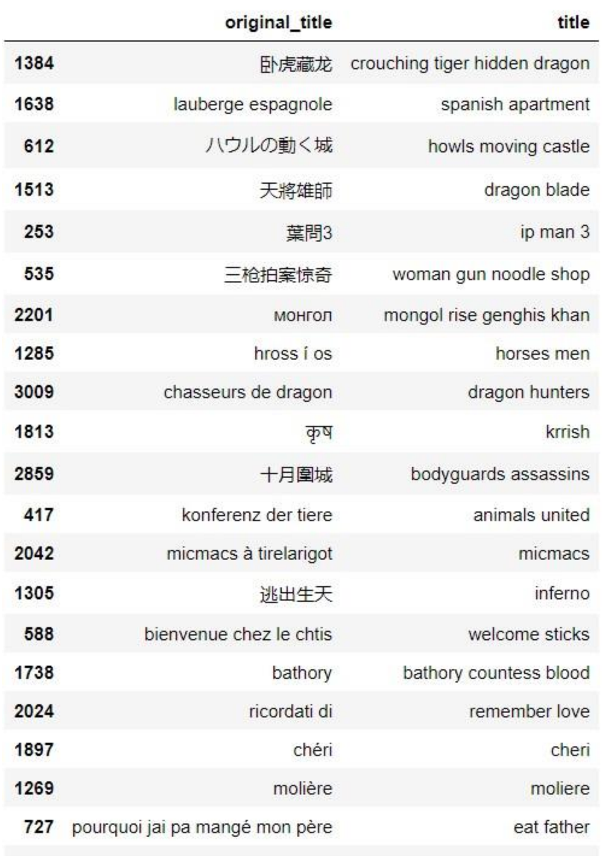
Preprocessing of the title column:

We applied lemmatization and We use TextBlob to obtain sentiment scores for each movie title. The sentiment score is computed using the polarity property of the TextBlob.sentiment object, which ranges from -1 to 1. A value closer to -1 indicates a negative sentiment, while a value closer to 1 indicates a positive sentiment.

The 'title' column of both the training and testing sets will be replaced with the corresponding sentiment score values.

Preprocessing of the Original Title column:

- We applied three preprocessing techniques to the original title column of the dataset: lemmatization, stop words removal, and punctuation removal. These steps helped to standardize the text, reduce noise, and improve model performance by removing unnecessary words and characters from the data.
- In this case, we have identified rows where the original title differs from the title. We then created a binary variable to indicate whether the two titles match or not. Specifically, if the titles are the same, we assign a value of 1 to the original title column. If they are not the same, we assign a value of 0 to the original title column. This process helps to distinguish between cases where there may have been modifications made to the original title and cases where the original title was used without alteration.
- Upon performing the summation of binary values assigned to the original title column, it has been observed that out of a total of 3040 rows, 2901 rows have been assigned a value of 1. This indicates that in these cases, the original title matches the title. On the other hand, the remaining 139 rows have been assigned a value of 0, indicating that most of these titles are **the same but in a different language**. As a result, we have decided to drop the original title column from the dataset.



	original_title	title
1384	卧虎藏龙	crouching tiger hidden dragon
1638	lauberge espagnole	spanish apartment
612	ハウルの動く城	howls moving castle
1513	天將雄師	dragon blade
253	葉問3	ip man 3
535	三枪拍案惊奇	woman gun noodle shop
2201	монгол	mongol rise genghis khan
1285	hross í os	horses men
3009	chasseurs de dragon	dragon hunters
1813	कृष	krrish
2859	十月圍城	bodyguards assassins
417	konferenz der tiere	animals united
2042	micmacs à tirelarigot	micmacs
1305	逃出生天	inferno
588	bienvenue chez le chtis	welcome sticks
1738	bathory	bathory countess blood
2024	ricordati di	remember love
1897	chéri	cheri
1269	molière	moliere
727	pourquoi jai pa mangé mon père	eat father

Preprocessing of the tageline column:

- We apply the sentiment analysis function 'get_sentiment_score' to the 'tagline' column of both the training and testing datasets. the function assigns a numerical score to each tagline in the dataset based on its overall sentiment (e.g. positive, negative, neutral). This score can then be used as a feature in machine learning models or other analyses.

Handling Columns in JSON Form:

- Certain columns contained data in JSON form (an array of dictionaries). The relevant information such as genre names, keywords, production companies, countries, and spoken languages were extracted from the dataset. This was achieved by excluding their respective IDs.

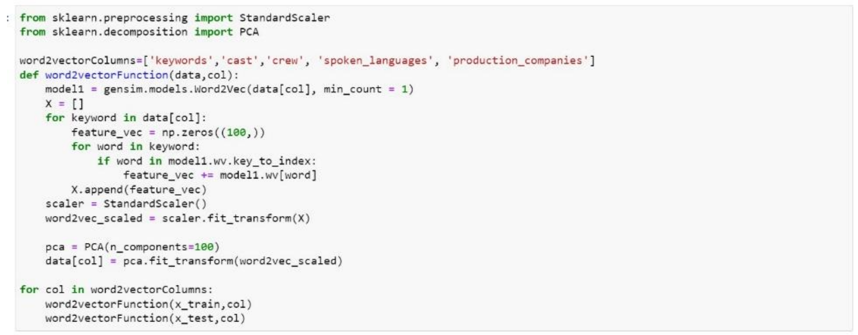
One hot encoding:

- One hot encoding was applied to two columns, namely genres and production countries. To accomplish this, a function was implemented to obtain the new columns. These new columns were saved to be added to X_test. We used the fit and transform functions with X_train and the transform function only with X_test.

Word2vector:

- We attempted to use a method for word-to-vector conversion in our machine learning model by leveraging the Python library scikit-learn's StandardScaler and PCA classes. The code used the Word2Vec algorithm from the gensim library to create feature vectors for columns containing text data (keywords, cast, crew, spoken_languages, production_companies) in the training and testing datasets.

- Initially, we implemented a code that scaled the feature vectors using StandardScaler before applying PCA on them. However, this resulted in reduced accuracy compared to not scaling the feature vectors.



```python
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA

word2vectorColumns=['keywords','cast','crew', 'spoken_languages', 'production_companies']
def word2vectorFunction(data,col):
    model1 = gensim.models.Word2Vec(data[col], min_count = 1)
    X = []
    for keyword in data[col]:
        feature_vec = np.zeros((100,))
        for word in keyword:
            if word in model1.wv.key_to_index:
                feature_vec += model1.wv[word]
        X.append(feature_vec)
    scaler = StandardScaler()
    word2vec_scaled = scaler.fit_transform(X)

    pca = PCA(n_components=100)
    data[col] = pca.fit_transform(word2vec_scaled)

for col in word2vectorColumns:
    word2vectorFunction(x_train,col)
    word2vectorFunction(x_test,col)
```

- Therefore, word2vector algorithm was applied,and matrix containing the results was obtained. Each row was replaced by its corresponding column, and the mean was calculated and replaced with it.
- Overall, we have successfully incorporated the word-to-vector technique in our machine learning model, and after experimentation with different methods, we have arrived at an optimal solution that yields improved performance.

Scaling:

- To standardize the numeric columns of X_train ('budget', 'viewer count', 'revenue', 'runtime', 'vote count', 'year', 'month', 'day'), Standard Scaler was applied (fit and transform). This scaling was then used on X_test (transform only).

Feature Selection:

In this section, we will describe the feature selection methods that we used to identify the most relevant features for our regression model.

Pearson's Correlation Coefficient

Pearson's correlation coefficient is a statistical measure that quantifies the linear relationship between two variables. It can be used to evaluate the correlation between each input feature and the target variable in a dataset. Features with a high correlation coefficient (positive or negative) are more likely to be good predictors of the outcome and can be selected for the model. Pearson's correlation coefficient is a simple and easy-to-use method that can quickly identify the most important features for a model.

KBest

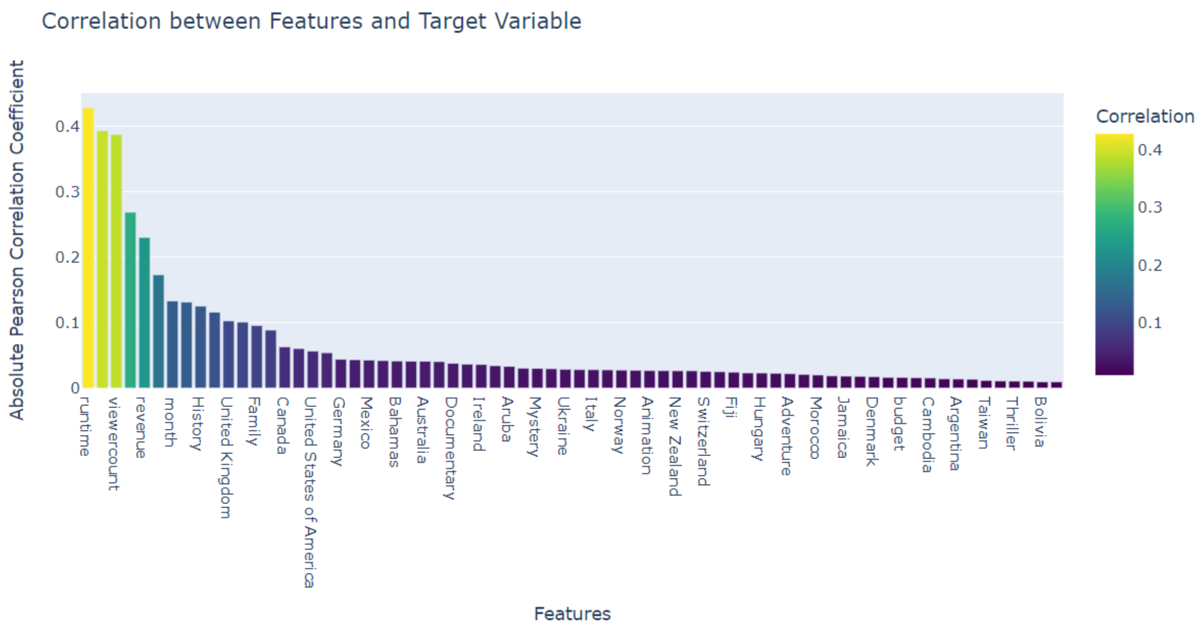
KBest is another feature selection method that selects the top K features based on their score on a given statistical test. For example, in the case of linear regression, the F-test is often used to evaluate the significance of each feature in predicting the outcome. KBest selects the K features with the highest F-test scores and discards the rest. This method can be used to identify features that are significant predictors of the outcome and can improve the model's performance.

Combining Pearson's Correlation Coefficient and KBest

We used both Pearson's correlation coefficient and KBest together to select the most relevant features for our regression model. This approach helped to ensure that the selected features were highly correlated with the target variable and were also statistically significant predictors of the outcome. This approach can improve the overall performance of the model and help to avoid overfitting, where the model may perform well on the training data but poorly on new, unseen data.

Selected Features

Based on the results of feature selection, the following features were chosen for the regression model:



Other features were discarded as they were found to have lower correlation with the target variable or were not deemed significant in predicting the target variable.

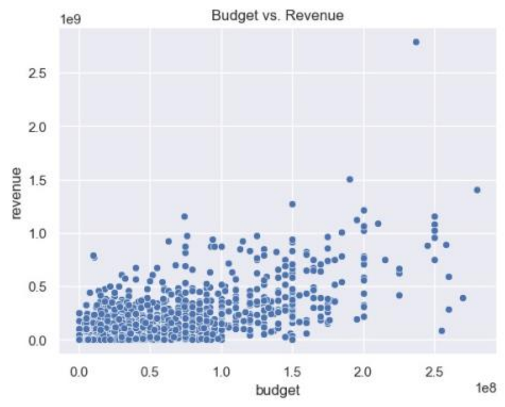
2. Dataset Analysis

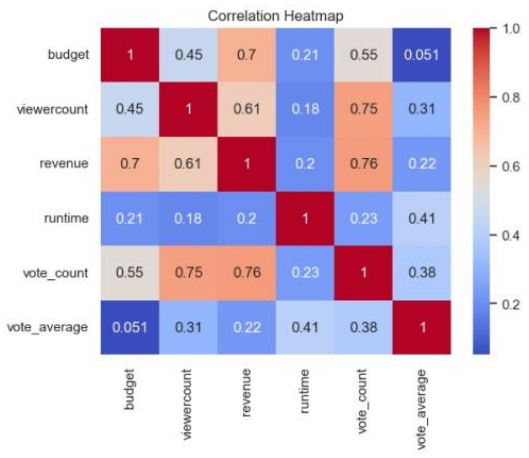
We performed an analysis of the dataset to gain a better understanding of the relationships between the variables. We used scatter plots to visualize the relationship between the budget and revenue columns, and found a positive correlation between the two variables, suggesting that higher budgets may lead to higher revenues.

We also created a heatmap to visualize the correlations between the numeric columns, including budget, viewer count, revenue, runtime, vote count, and vote average.

The heatmap showed strong positive correlations between revenue and budget, as well as between vote count and revenue, suggesting that these variables may be important predictors of movie success.

These findings provide important insights into the factors that drive movie revenues and can inform future movie production and marketing strategies.





	budget	viewercount	revenue	runtime	vote_count	vote_average
budget	1	0.45	0.7	0.21	0.55	0.051
viewercount	0.45	1	0.61	0.18	0.75	0.31
revenue	0.7	0.61	1	0.2	0.76	0.22
runtime	0.21	0.18	0.2	1	0.23	0.41
vote_count	0.55	0.75	0.76	0.23	1	0.38
vote_average	0.051	0.31	0.22	0.41	0.38	1

3. Models:

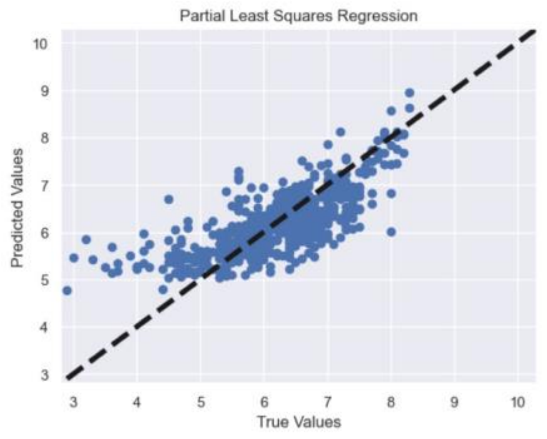
1) **Support Vector Regression (SVR)**: Using a radial basis function kernel and hyperparameters C=1.9999 and gamma='scale', SVR achieved an **RMSE of 0.58** and an **R^2 score of 0.60**. SVR is a type of regression that uses support vector machines (SVMs) to perform regression tasks. It can handle non-linear relationships between the

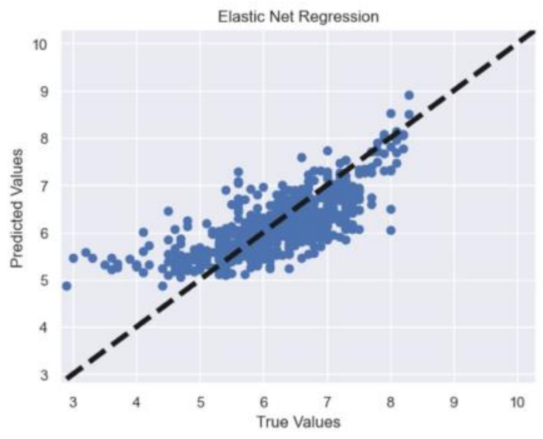
predictor variables and the outcome variable, but it can be sensitive to the choice of kernel function and regularization parameters.

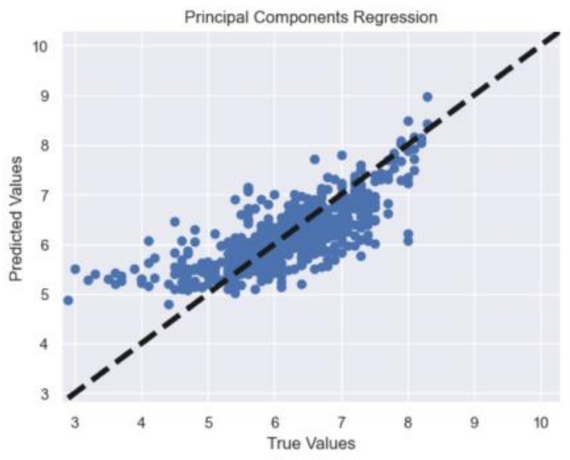
2) **Ridge Regression**: This model achieved an **RMSE of 0.61** and an **R^2 score of 0.56**. It used hyperparameters alpha=10.6 and solver=`'sparse_cg'`. Ridge Regression is a type of linear regression that uses L2 regularization to prevent overfitting. Its performance was moderate in this case.
3) **Principal Components Regression (PCR)**: After performing PCA on the data and training a Linear Regression model on the transformed data, PCR achieved an **RMSE of 0.6** and **R-squared score of 0.56**. PCR is a technique for reducing the dimensionality of a dataset by projecting the data into a lower-dimensional space.
4) **Elastic Net Regression**: With hyperparameters alpha=0.01 and l1_ratio=0.0, Elastic Net Regression achieved an **RMSE of 0.6** and **R-squared score of 0.56**. Elastic Net is a combination of Lasso and Ridge regressions and can handle multicollinearity among predictors.
5) **Partial Least Squares Regression (PLS)**: PLS Regression, with n_components = 5, achieved an **RMSE of 0.615** and **R-squared score of 0.55**. PLS is another technique for reducing the dimensionality of a dataset and is often used when there are many features and multicollinearity is present.
6) **Polynomial Regression**: By creating polynomial features using Polynomial Features(degree=1), and then fitting a Linear Regression model, Polynomial Regression achieved an **RMSE of 0.62** and an **R-squared score of 0.55**. Polynomial Regression is a form of regression analysis in which the relationship between the independent variable x and the dependent variable y is modeled as an nth degree polynomial.

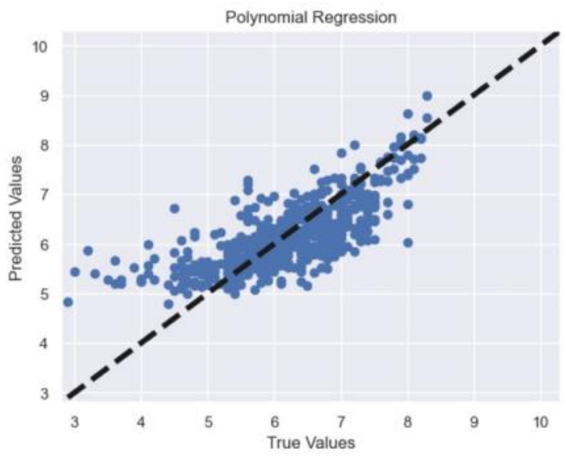
Overall, SVR had the highest accuracy with an **R-squared score of 0.60**.

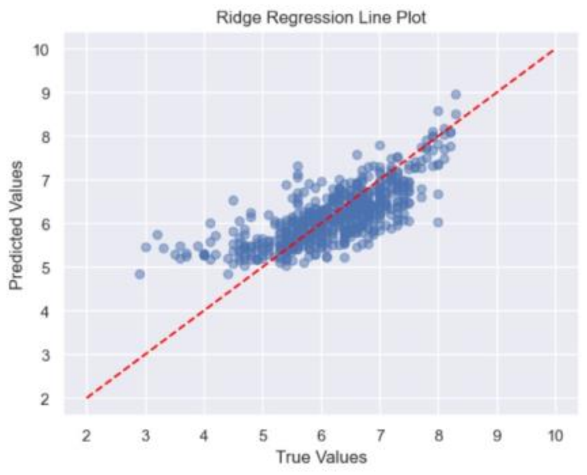
4. Resultants regression line plots

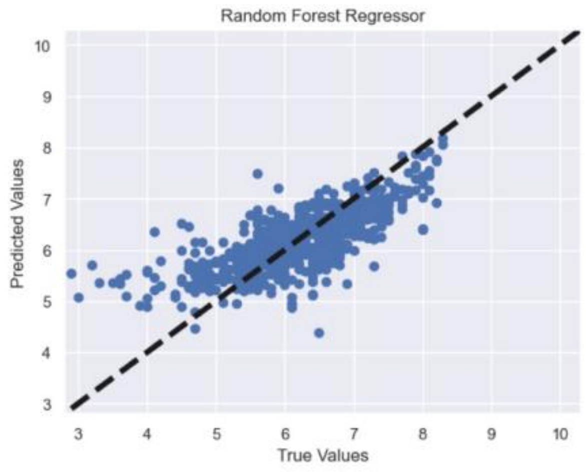


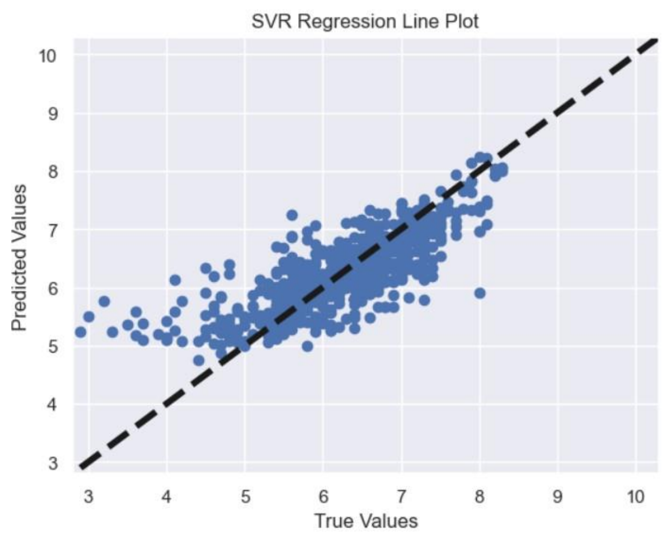












5. Techniques used to improve results:

- **Hyperparameter Tuning Using GridSearchCV:**
  It helps to loop through predefined hyperparameters and fit the model on the training set, So that we can select the best hyperparameters of the model from listed hyperparameters.

6. conclusion:

In this phase of the machine learning movie project, we focused on data preprocessing, handling missing values, detecting, and handling outliers, scaling, and feature engineering.

First, we gathered the required data by merging the "movies" and "credits" datasets. We then split the data into training and testing sets.

Next, we addressed missing values by dropping rows with missing runtime values and replacing zeros in the budget and runtime columns with their respective means.

To handle outliers, we visually inspected boxplots and applied a limit-based approach using the interquartile range (IQR). Outliers beyond the upper and lower limits were replaced with the respective limits.

The data was scaled using the StandardScaler to ensure that all features were on a similar scale, which is important for many machine learning algorithms.

We also performed pre-processing on specific columns such as "title," "original_language," and various JSON columns like "genres" and "production_countries." Techniques like label encoding and one-hot encoding were used to convert categorical data into numerical representations.

Additionally, we utilized word2vec embeddings to extract meaningful features from text-based columns such as "keywords," "cast," and "crew." We calculated the mean values for these columns to further enhance the feature representation.

After feature selection using the SelectKBest algorithm, we narrowed down the features for the regression models.

Finally, we implemented:

1) Support Vector Regression (SVR) model: **R^2 score of 0.60**
2) Ridge Regression: **R^2 score of 0.56**
3) Principal Components Regression (PCR): **R-squared score of 0.56**
4) Elastic Net Regression: **R-squared score of 0.56**
5) Partial Least Squares Regression (PLS): **R-squared score of 0.55**
6) Polynomial Regression: **R-squared score of 0.55**

Overall, this phase of the project involved comprehensive data pre-processing, outlier handling, feature engineering, and model training. The results obtained from the SVR model provided insights into predicting movie ratings based on various factors. Further iterations and improvements can be made to enhance the model's performance and predictive capabilities.