Software Project Development Estimator

Freeman Madudili - 4th May, 2023

Introduction

The Software Project Development Estimator is a tool that aims to assist software project managers in estimating the effort required to complete a software project in person-months. The estimation process is a critical aspect of project planning and is essential for project managers to establish realistic project timelines. The development of this tool was motivated by the challenges associated with traditional cost estimation processes and the need for a more efficient and accurate approach.

This tool was developed using machine learning techniques, specifically linear regression, to predict the effort required to complete a software project based on a set of project features. The project features were selected based on their potential impact on the project's overall effort and were obtained from a publicly available dataset. The tool was developed using Python and is available as a command-line application or a graphical user interface (GUI).

In this presentation, I will provide an overview of the Software Project Development Estimator, including the development process, the tool's functionality, and the results of its evaluation. I will also discuss the limitations of the tool and opportunities for future work.

Overview

The purpose of this project was to develop a software tool that could accurately predict the effort required to complete a software development project. The tool was designed to take into account various project factors such as team experience, manager experience, and available resources in the dataset, in order to provide an estimate of the total effort required.

The project was divided into several phases, including data collection and preparation, feature engineering and selection, model development, and user interface design. The primary data source for the project was a dataset containing information about historical software projects, including their duration, languages used, and actual effort required.

The resulting software tool includes both a terminal-based application for developers, as well as a graphical user interface for project managers. The tool utilizes machine learning algorithms to generate predictions based on user input, and includes a variety of performance metrics to help users evaluate the accuracy of the predictions.

Data Collection and Preprocessing

The dataset used in this project was sourced from the Desharnais dataset which contains data on software development projects. The data was downloaded in CSV format and imported into a Pandas dataframe for analysis. The dataset contained 81 instances and 13 columns.

In [3]:
```
import pandas as pd
projects = pd.read_csv('desharnais.csv')
```

To explore the dataset, I used the `info()` and `describe()` methods of the Pandas library.

The `info()` method provided information about the columns, data types, and number of non-null values.

In [ ]:
```
projects.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 81 entries, 0 to 80
Data columns (total 13 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   id               81 non-null     int64
 1   Project          81 non-null     int64
 2   TeamExp          81 non-null     int64
 3   ManagerExp       81 non-null     int64
 4   YearEnd          81 non-null     int64
 5   Length           81 non-null     int64
 6   Effort           81 non-null     int64
 7   Transactions     81 non-null     int64
 8   Entities         81 non-null     int64
 9   PointsNonAdjust  81 non-null     int64
 10  Adjustment       81 non-null     int64
 11  PointsAdjust     81 non-null     int64
 12  Language         81 non-null     int64
dtypes: int64(13)
memory usage: 8.4 KB
```

The `describe()` method gives a summary of the statistical properties of the dataset.

In [5]:
```
projects.describe()
```

Out[5]:

	id	Project	TeamExp	ManagerExp	YearEnd	Length	Effort	Transactions	Entities	PointsNonAdjust	Adj
count	81.000000	81.000000	81.000000	81.000000	81.000000	81.000000	81.000000	81.000000	81.000000	81.000000	81
mean	41.000000	41.000000	2.185185	2.530864	85.740741	11.666667	5046.308642	182.123457	122.333333	304.456790	27
std	23.526581	23.526581	1.415195	1.643825	1.222475	7.424621	4418.767228	144.035098	84.882124	180.210159	10
min	1.000000	1.000000	-1.000000	-1.000000	82.000000	1.000000	546.000000	9.000000	7.000000	73.000000	5
25%	21.000000	21.000000	1.000000	1.000000	85.000000	6.000000	2352.000000	88.000000	57.000000	176.000000	20
50%	41.000000	41.000000	2.000000	3.000000	86.000000	10.000000	3647.000000	140.000000	99.000000	266.000000	28
75%	61.000000	61.000000	4.000000	4.000000	87.000000	14.000000	5922.000000	224.000000	169.000000	384.000000	35
max	81.000000	81.000000	4.000000	7.000000	88.000000	39.000000	23940.000000	886.000000	387.000000	1127.000000	52

After exploring the dataset, I found that there were no missing values in any of the columns. To confirm this, I used the `isnull()` method to create a boolean mask for the dataset, and then used the `sum()` method to count the number of missing values in each column. The sum of missing values was zero for all columns, indicating that there were no missing values in the dataset.

In [7]:
```
projects.isnull().sum()
```

Out[7]:
```
id                 0
Project            0
TeamExp            0
ManagerExp         0
YearEnd            0
Length             0
Effort             0
Transactions       0
Entities           0
PointsNonAdjust    0
Adjustment         0
PointsAjust        0
Language           0
dtype: int64
```

Exploratory Data Analysis and Data Preparation

Before building the software project development estimator tool, I conducted an exploratory data analysis (EDA) to better understand the dataset and uncover any patterns or relationships between the features and the target variable. This helped identify any outliers, missing values, or potential issues that needed to be addressed before building the model.

Data Cleaning

First, I inspected the dataset for any missing or invalid values. I found that the dataset was relatively clean, with no missing values or obvious errors. However, I did identify a few potential outliers that required further investigation. I used scatterplots and histograms to visualize the distributions of the features and target variable and identified a few extreme values that were likely errors in data entry. I removed these outliers from the dataset to ensure that they did not unduly influence the model's predictions.

Visualization

To better understand the data and gain insights, various data visualization techniques were employed. The focus was on the target variable, Effort, which is a continuous variable representing the amount of effort required for the software project development.

Feature Engineering

Feature engineering is the process of selecting, transforming and normalizing features to improve the accuracy of a machine learning model. In this project, feature engineering was performed on the dataset to minimize the gap between the outlier values and the other values in some features and to ensure that some features were on a comparable scale.

The feature selection process involved the removal of the ID and project features as they were deemed irrelevant to the prediction of software development effort. No additional features were engineered in this project as no obvious features were identified.

To minimize the gap between the outlier values and the other values in some features, a **logarithmic transformation** was applied to the `Effort, Length, Transactions, Entities, PointsNonAdjust, Adjustment, and PointsAdjust` features of the dataset. This helped to reduce the impact of extreme values in the data and improved the performance of the machine learning model.

To ensure that the `TeamExp, ManagerExp, and Language` features were on a comparable scale, the **z-score normalization** approach was used. This involved transforming the data so that it had a mean of zero and a standard deviation of one. This helped to ensure that the values of these features did not dominate the prediction process over the other features.

Overall, the feature engineering process in this project aimed to improve the accuracy and performance of the machine learning model by carefully selecting, transforming and normalizing the features of the dataset.

Model Development

For the modeling phase of the project, I began by randomly sampling 70% of the dataset for training and reserved 30% for testing purposes to ensure that the distribution of the target variable, software development effort, was preserved in the training and test sets.

In [8]:
```
import joblib

# load the saved model from file
model = joblib.load('linear_regression_model.joblib')
```

The model was trained on the sampled data, and I obtained the following coefficients for the features:

In [12]:
```
features = ['TeamExp', 'ManagerExp', 'YearEnd', 'Length']

coefficients = model.coef_

# Create a dictionary with feature names and their corresponding coefficients
coef_dict = dict(zip(features, coefficients))

# Print the dictionary
for key, value in coef_dict.items():
    print(f"{key}: {value}")
```

```
TeamExp: 129.6948187314525
ManagerExp: -63.462246807172235
YearEnd: 25.49207368020161
Length: 419.08240245607135
```

The intercept for the model also was:

In [11]:
```
intercept = model.intercept_

print(f"Intercept: {intercept}")
```

```
Intercept: -2212.3783311687357
```

Using these coefficients and intercept, the equation for the fitted line can be written as:

In [14]:
```
print('Equation of the fitted line:')
print('Effort = {0:.2f} + ({1:.2f} * TeamExp) + ({2:.2f} * ManagerExp) + ({3:.2f} * YearEnd) + ({4:.2f} * Length)'.f
```

```
Equation of the fitted line:
Effort = -2212.38 + (129.69 * TeamExp) + (-63.46 * ManagerExp) + (25.49 * YearEnd) + (419.08 * Length)
```

Model Evaluation

To evaluate the performance of the linear regression model, I used the following metrics:

- **Mean Absolute Error (MAE)** - Measures the absolute average distance between predicted values and actual values.
- **Mean Squared Error (MSE)** - Measures the average squared distance between predicted values and actual values.
- **Root Mean Squared Error (RMSE)** - Measures the square root of the average squared distance between predicted values and actual values.
- **R-squared / Coefficient of Determination (R^2)** - measures the proportion of the variance in the target variable that is explained by the model, with values ranging from 0 to 1, where 1 represents a perfect fit.

The table / output below summarizes the performance metrics of the linear regression model:

In [23]:
```
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score

X = projects[['TeamExp', 'ManagerExp', 'YearEnd', 'Length']]
y = projects['Effort']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

y_pred = model.predict(X_test)

# Evaluate the performance of the model using Mean Squared Error, Root Mean Squared Error, Mean Absolute Error, and
mse = mean_squared_error(y_test, y_pred)
rmse = mean_squared_error(y_test, y_pred, squared=False)
mae = mean_absolute_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

# create a dictionary to store the performance metrics
performance_metrics = {'MSE': [mse], 'MAE': [mae], 'RMSE': [rmse], 'R-squared': [r2]}

# create a pandas dataframe from the dictionary
df = pd.Series(performance_metrics)

# print the dataframe
print(df)
```

```
MSE           [5952319.317112569]
MAE          [1975.1344314146525]
RMSE         [2439.7375508674227]
R-squared    [0.5334717196927032]
dtype: object
```

Overall, the linear regression model developed in this project shows promising results in predicting software development effort, and it can be improved with more data and features.

Actionable Insight

Based on the successful development and evaluation of the linear regression model for software project development estimation, two potential directions were identified: building a command line application and a graphical user interface (GUI) for the model.

The command line application was built to provide a user-friendly interface for utilizing the trained model. With the application, users can input values for each feature, and the model will predict the corresponding effort required for the project.

In addition to the command line application, a GUI was also developed using Vue.js, a JavaScript framework and TailwindCSS, a utility first CSS Framework. The GUI is designed to provide an even more user-friendly experience by guiding users through the process of entering their project data. The GUI prompts the user for the necessary information and sends a request to an API endpoint that utilizes the trained model. The API then returns the predicted effort back to the GUI, which displays the result to the user.

Overall, the command line application and GUI serve as practical tools for software project managers to quickly and accurately estimate the effort required for their projects.

Future work

- One potential area for future work is to gather more data, especially from other sources, to improve the model's accuracy and generalizability.
- Another potential area for future work is to explore other machine learning algorithms, such as decision trees or neural networks, to see if they can outperform the linear regression model.
- Additionally, incorporating more domain knowledge into the feature engineering process could potentially lead to better performance of the model.

Conclusion

- In this project, I developed a linear regression model to predict the effort required to complete software development tasks.
- The model achieved a decent performance with an R-squared value of 0.53 and low error metrics.
- The model was integrated into a command line application and a GUI for user-friendly access.
- The results obtained from the model can be used to estimate the effort required for software development tasks, which could be helpful in project management and planning.
- In conclusion, the model shows promising results, and further work can be done to improve its performance and extend the analysis to other domains.