|  |  |
| --- | --- |
|  | Movies Revenue Prediction |
|  |  |
|  | CS\_13  Milestone 2  5/24/22 |

**Preprocessing:**

* **Filling missing directors** using TMDb API by matching the movie name and its release date with the search results to ensure getting the corresponding director, included in “director\_filling.py”.
* **Determining director popularity** using TMDb API and instead of using normal encoding techniques we replaced each director with his popularity, included in “director\_filling.py”.
* **Determining voice actors popularity** using TMDb API and instead of using normal encoding techniques we replaced each voice actor with his popularity, then calculated the average popularity of all the voices actors for each movie, then added a new feature to represent the voice actors popularity for each movie. Included in “director\_filling.py”
* **Format release date** to the universal format included in ”format\_date.py” then calculated the date into one number by the following formula:

Included in “calc\_date.py”.

* **Encoding the MPAA\_rating and genre** columns using dummies. Included in “rating\_genre\_preprocessing.py”.
* **Feature Engineering**: Adding two new features 'CharactersCount' which represents number of voice actors in the movie and 'IsAnimation' which represents the type of the movie. Included in “feature\_Engineering.py”.
* **Joining** ‘movie-revenue.csv’ table and ‘movie-director.csv’ table by using pandas merge method. Included in “join\_tables.py”.
* **Scaling features**: by using MinMaxScaler().Included in “classification\_models.py”.
* **Filling missing data:** by getting the mean of the columns that have NaN such as directors and Actors then we fill the NaN values with the mean of each column using fillna() function in python. Included in “fill\_empty\_cells.py”.

**Classification models:**

* **Decision tree model:**
  + **Training Time:** **0.00100255012512207**
  + **Testing Time: 0**
  + **Accuracy: 0.440860215053763**
* **RBF model:**
  + **Training Time:** **0.00800704956054687**
  + **Testing Time: 0.00400543212890625**
  + **Accuracy: 0.408602150537634**
* **Linear Svc model:**
  + **Training Time:** **0.00100040435791015**
  + **Testing Time: 0**
  + **Accuracy: 0.408602150537634**
* **Kernel Linear model:**
  + **Training Time:** **0.00500297546386718**
  + **Testing Time: 0.00100111961364746**
  + **Accuracy: 0.408602150537634**
* **Logistic Regression model:**
  + **Training Time:** **0.0200164318084716**
  + **Testing Time: 0**
  + **Accuracy: 0.301075268817204**
* **Polynomial SVC model:**
  + **Training Time:** **0.00606036186218261**
  + **Testing Time: 0.000945568084716796**
  + **Accuracy: 0.408602150537634**

**Hyper parameters tuning:**

* **For Logistic regression:** 
  + - : for **large** high bias, low variance (under fitting)
    - : for **small** low bias, high variance (over fitting)
* **For SVM:** 
  + - : for **small** high bias, low variance (under fitting) soft–margin classifier (more generalization).
    - : for **large** low bias, high variance (over fitting). hard–margin classifier (best accuracy).
    - Kernel functions:
      * Linear kernel.
      * The polynomial kernel.
      * Gaussian RBF.
* **For decision tree:** 
  + - max depth: large depth will get large training accuracy but will lead to overfitting, bad generalization and high variance (overfitting).
    - max depth: small depth will get bad training accuracy , bad generalization and high bias (under fitting).
    - * Equal zero: standard decision tree learning
      * Equal infinity: only root with y’ is the majority class
      * In between: Balance fit and complexity

**Hyper parameters trials:**

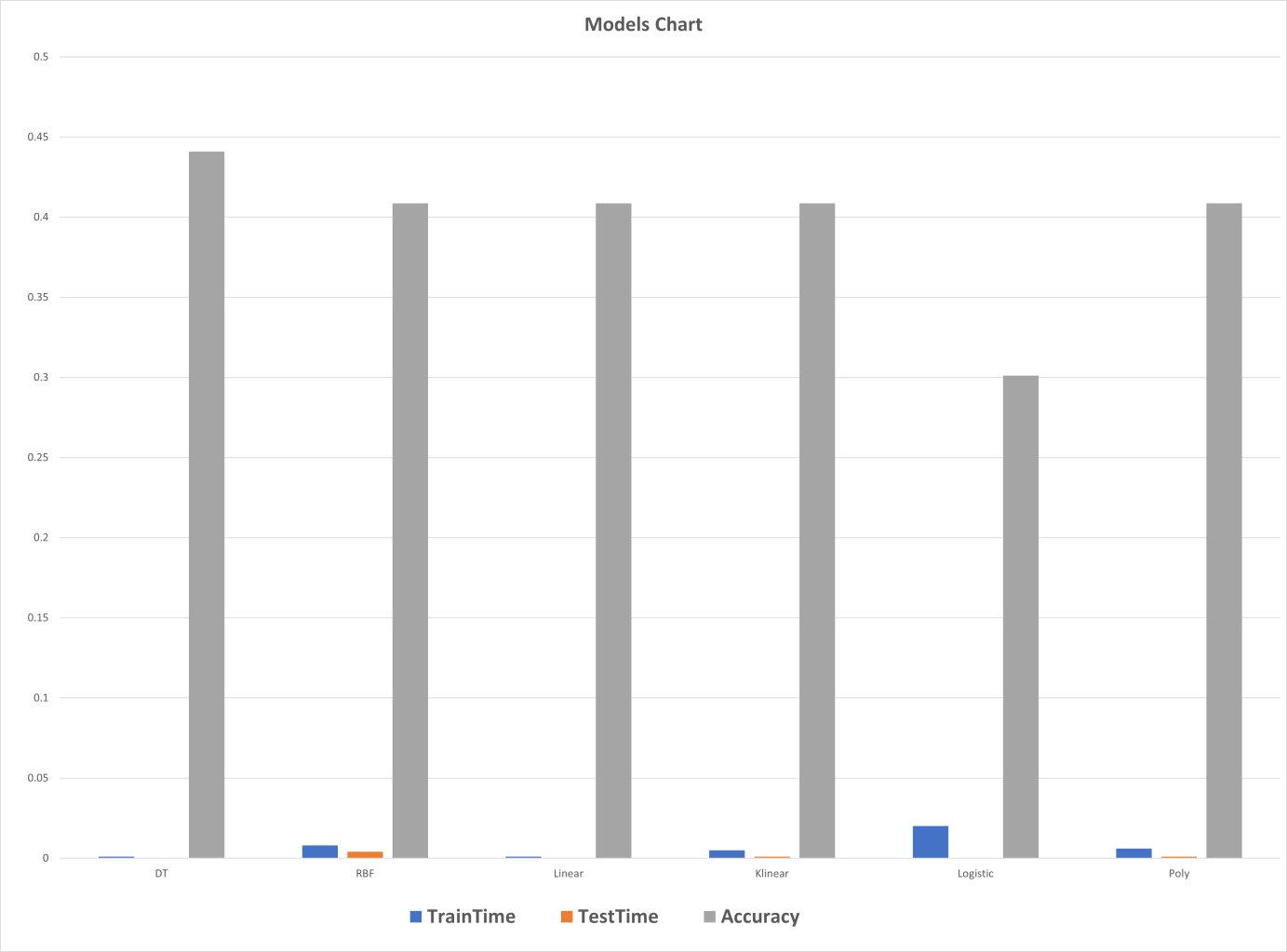
* **C:** hyper parameters in SVM: the kernel will be Linear Kernel

|  |  |
| --- | --- |
| **C value** | **Accuracy** |
| 0.00001 | 0.40860215053763443 |
| 0.1 | 0.3655913978494624 |
| 1.2 | 0.3118279569892473 |

* **Degree in polynomial SVM model:** C will be constant at 0.00001

|  |  |
| --- | --- |
| **Degree** | **Accuracy** |
| 20 | 0.41935483870967744 |
| 7 | 0.40860215053763443 |
| 100 | 0.26881720430107525 |

**Three bar graph:**

****

**Conclusion:**

For phase 2, we can't say we had much intuition about what the output would be or what should we expect. After yielding an MSE that would barely fit into the size of a 64-bit integer in phase 1, we proceeded in phase 2 with caution.

With expectations that low we weren't too shocked to see that no classification model could hit accuracy above 45 percent, the data is simply insufficient to produce a higher accuracy.

Our preprocessing didn't differ from phase 1; just one thing that we elected not to do: To not extract top features.

We concluded that most features were extracted via dummies and in the first place they were originally a few, so we tried to feed the models all the features and it didn't disappoint, on the contrary, it gave us better accuracies, especially the decision tree model.

If we learned anything from this project, it would be that the data itself is the most important thing about machine learning, these rows of raw data that we collect and process to work with.

Also that the preprocessing phase is the most crucial to this process, the choices we make about the data, how should we represent every piece of it, trying multiple things, and going through endless possibilities to get the best possible model.

To sum it up, our intuition going into phase 2 was that we were going to have to do our best to squeeze that very few rows into good training data for our models to work properly, we feared that it wouldn't be sufficient and our fears were true.