Milestone 2 Report (Predict Movie Success)

- **Intro:**
  For classification: Model used are Logistic Regression, SVM, AdaBoost KNN and Gradient Boosting. Best feature selection all features expect ['title', 'age'].

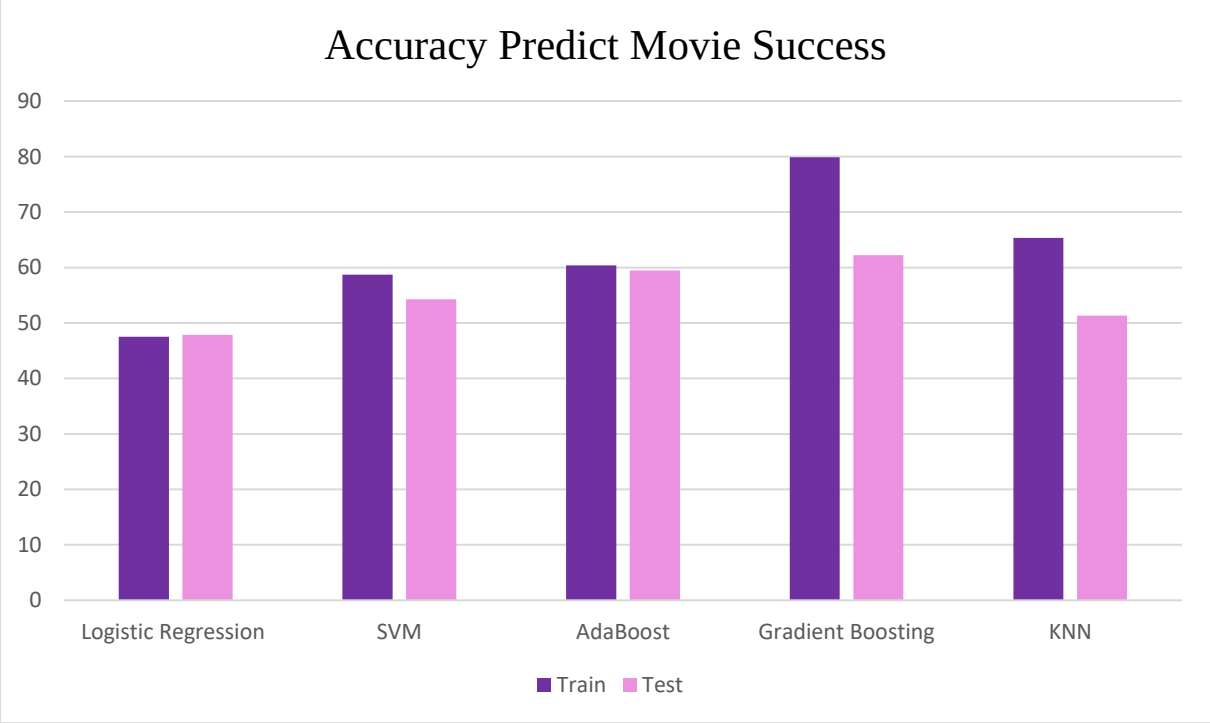
1- Bar Graph:

1. **<u>Accuracy (bar chart)</u>**
   Logistic Regression train → 47.52
   Logistic Regression test → 47.87

   SVM train → 58.68
   SVM test → 54.29

   AdaBoost train → 60.38
   AdaBoost test → 59.48

   Gradient Boosting train → 79.85
   Gradient Boosting test → 62.21
   KNN train → 65.31
   KNN test →51.32

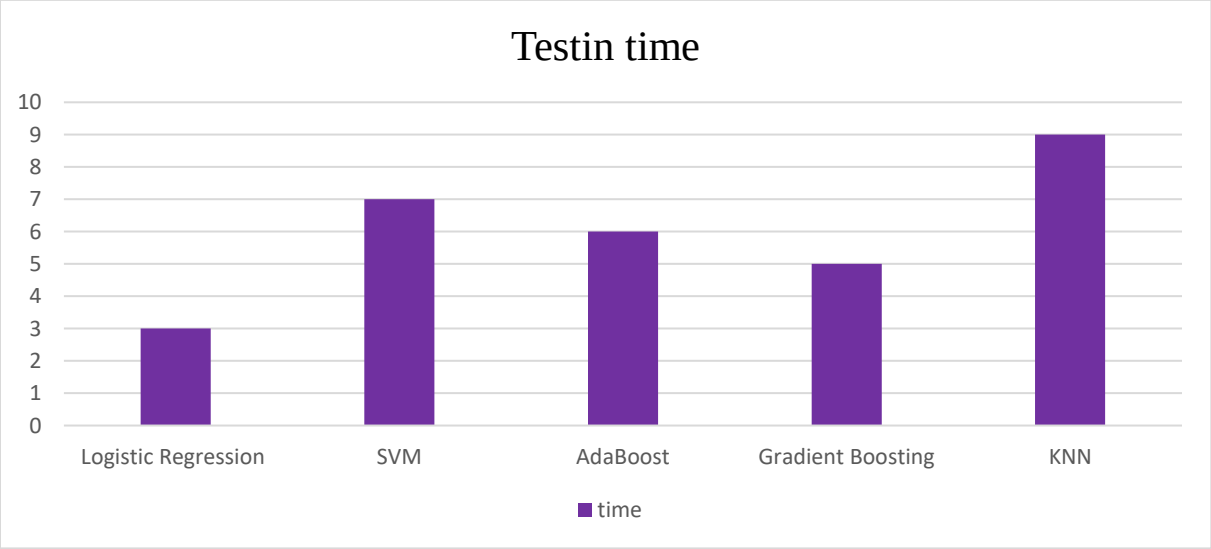


2. **Training time (bar chart)**

Logistic Regression train → 5 sec
SVM train → 4 mins (240 sec)
AdaBoost train → 24 sec
Gradient Boosting train → 1 min 8 sec (68 sec)
KNN → 30 sec

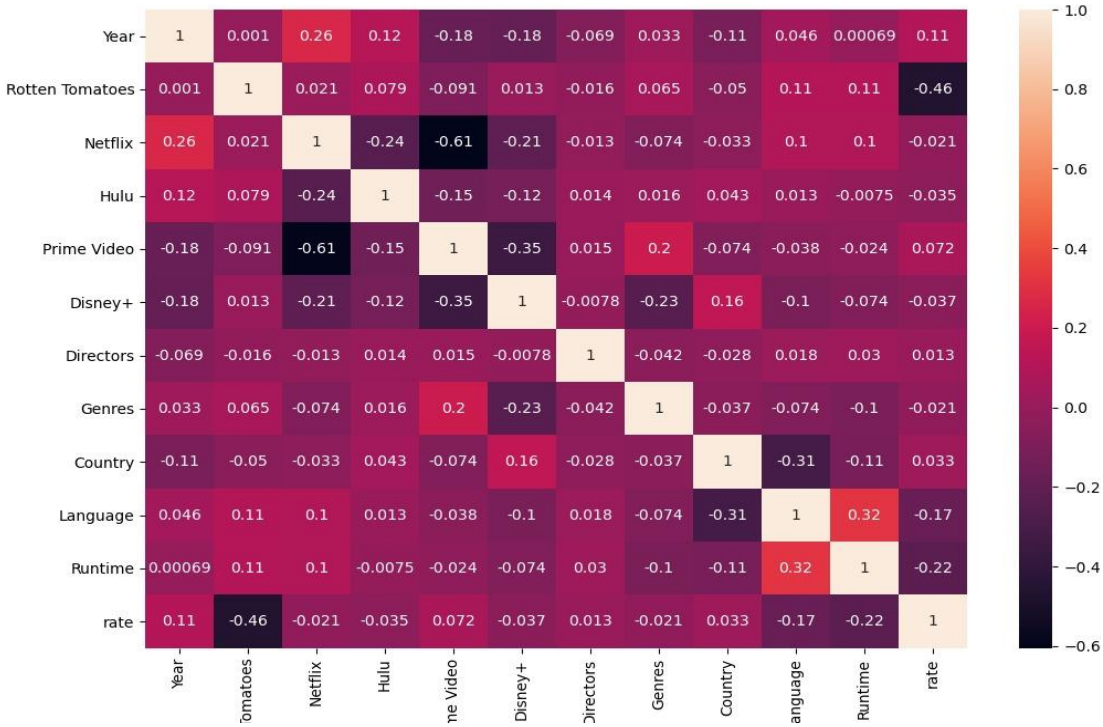
3. **Test time (bar chart)**

Logistic regression test → 3 sec
SVM test → 7 sec
AdaBoost test → 6 sec
Gradient Boosting test → 5 sec
KNN → 9 sec



2- Feature Selection:

Dropped columns because it has the least correlation with rate.



- Used all expect title:
  - Best accuracy gradient boosting train → 79.93
  - Best accuracy gradient boosting test → 60.04

- Used ['year', 'age', 'Rotten Tomatoes', 'Genres', 'Country', 'Language', 'Runtime'].
  - Best accuracy gradient boosting train → 72.39
  - Best accuracy gradient boosting test → 61.02

- Used all expect title, age, Netflix, Hulu, prime, video and Disney+.
  - Best accuracy gradient boosting train → 79.54
  - Best accuracy gradient boosting test → 61.50

- Used all expect title and age.
  - Best accuracy gradient boosting train → 79.85
  - Best accuracy gradient boosting test → 62.21

3- Hyperparameter tuning:

1. SVM

a. Kernel

I. Linear (1 VS all) → it was the simplest and fastest kernel but didn't deliver a good accuracy
II. RBF (Radial Basis Function) → it is more complex than the Linear one and so it delivered a slightly more accurate result
III. Polynomial (degree 6) → it is the most complex of them and delivered the most accurate results compared to the other 2. Higher degree delivered better results but costs more (time)

b. C (inverse of regularization parameter)

I. Small (0.001) → the smaller it is the lower variance the model is and so delivered bad accuracy
II. Large (100) → the larger it is the higher variance the model is and so the test accuracy is bad
III. Moderate (1) → it is in the middle of the previous 2 and delivered a perfect balance so the model won't overfit or underfit.

2. Logistic Regression

a. C (inverse of regularization parameter)

I. Small (0.001) → the smaller it is the lower variance the model is and so delivered bad accuracy
II. Large (100) → the larger it is the higher variance the model is and so the test accuracy is bad
III. Moderate (1) → it is in the middle of the previous 2 and delivered a perfect balance so the model won't overfit or underfit.

b. Solver/optimizer algorithm:

I. 'sag' and 'saga' are faster for large data set. Saga was used because it supports elastic net (regression).
II. 'liblinear' is a good choice for small data set.
III. 'lbfgs', 'sag' and 'saga' handle L2.

- Saga and LBFGS delivered almost the same accuracy (Saga is slightly higher), but liblinear delivered the lowest accuracy.

3. Decision tree with Adaboost

a. No. of estimators (with default learning rate 1)

I. Small (10) → it was the fastest and the accuracy wasn't so bad.
II. Large (1000) → it was the slowest and overfits most of the time.
III. Moderate (100) → it delivered the best accuracy among the three but not the best among the 6.

b. Learning rate

I. Small (0.01) → is useful when using a large number of estimators or large depth (unlikely to cause overfitting)
II. Large (1.4) → is useful when using a small number of estimators or small depth (as it is more likely to cause overfitting very easily)
III. Moderate (0.17) → it delivered the best accuracy

4. Gradient Boosting

a. No. of estimators

I. Small (10) → it was the fastest and the accuracy wasn't so bad
II. Large (1000) →it was the slowest and overfits most of the time
III. Moderate (100) → it delivered the best accuracy among the 6

b. Learning rate

I. Small (0.01) → is useful when using a large number of estimators or large depth (unlikely to cause overfitting)
II. Large (1.4) →is useful when using a small number of estimators or small depth (as it is more likely to cause overfitting very easily)
III. Moderate (0.17) → it delivered the best accuracy among the 3

5. KNN:

a. K-neighbors (3-9):

I. Small → it was the fastest and the accuracy wasn't so bad
II. Moderate → it delivered the best accuracy
III. Large →it was the slowest and overfits most of the time

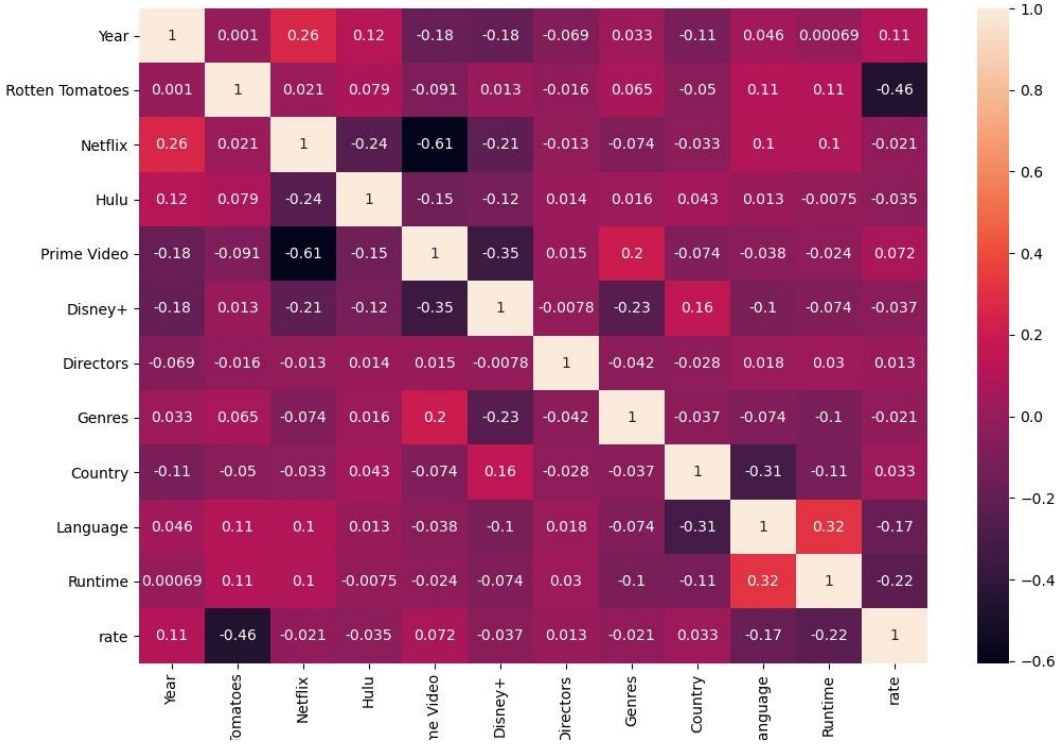
b. Weight:

I. Large k with distance →overfitting.
II. Small k with distance → accuracy wasn't so bad
III. uniform → accuracy wasn't so bad overfitting happens rarely.

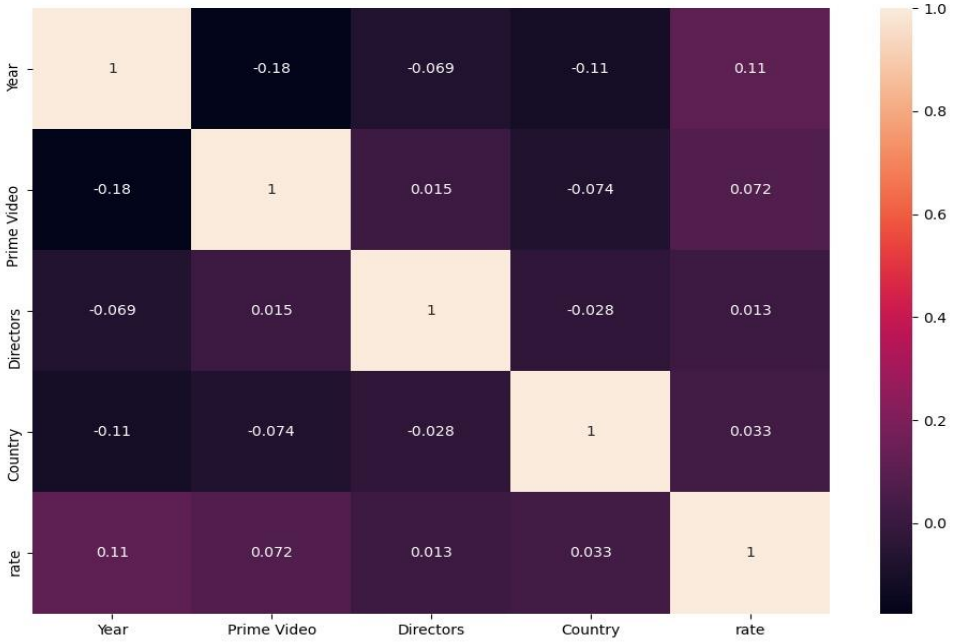
4- Conclusion:

1. Correlation between features:

➢ All features with rate:



➢ Features greater than 0:



2. **Preprocessing:**
   - Drop columns that does not affected in result:
     - Title, Netflix, Hulu, Prime Video, Disney+, Type and Directors.
   - solve missing data by:
     - interpolate for ('age', 'Rotten Tomatoes').
   - Encoding for categorical data:
     - ['age', 'Rotten Tomatoes', 'Genres', 'Country', 'Language', 'Runtime', 'rate'].
   - Normalization or Standardization.
     - Standardization used when all data around the mean.
     - Slight difference between them
     - Using standardization for AdaBoost (Very slightly higher than normalization in accuracy).
   - Split data 80% for training and 20% for validation.

3. **models:**
   - **Logistic regression:**
     - designed for this purpose (classification) and is most useful for understanding the influence of several independent variables on a single outcome variable.
     - Works only when the predicted variable is binary.
   - **Gradient Boosting** (is the best accuracy with 62.21%):
     - Efficiency and ease of implementation.
     - Requires a number of hyper-parameters and it is sensitive to feature scaling.
   - **AdaBoost Decision tree**:
     - can handle both numerical and categorical data.
     - Decision tree can create complex trees that do not generalize well.
   - **SVM:**
     - Effective in high dimensional spaces and uses a subset of training points in the decision function so it is also memory efficient.
     - The algorithm does not directly provide probability estimates, these are calculated using an expensive five-fold cross-validation.

- **KNN:**
  - simple to implement, robust to noisy training data, and effective if training data is large.
  - takes a lot of time (computation cost is high).

4. Accuracy:

Training:

Gradient Boosting > KNN > AdaBoost > SVM > Logistic Regression.

Testing:

Gradient Boosting > AdaBoost > SVM > KNN > Logistic Regression.