**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:**
2. **Accuracy (bar chart)**

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

1. **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

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

Logistic regression test → 3 sec

SVM test → 7 sec

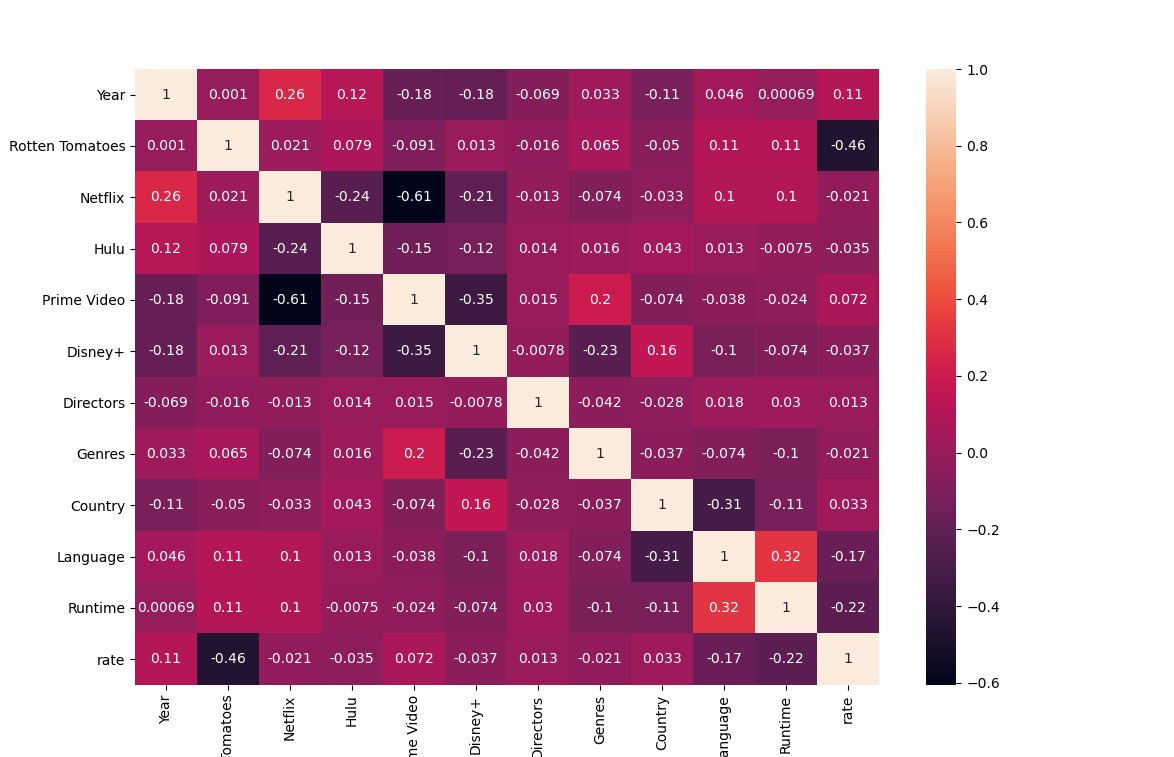
AdaBoost test → 6 sec

Gradient Boosting test → 5 sec

KNN 🡪 9 sec

1. **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

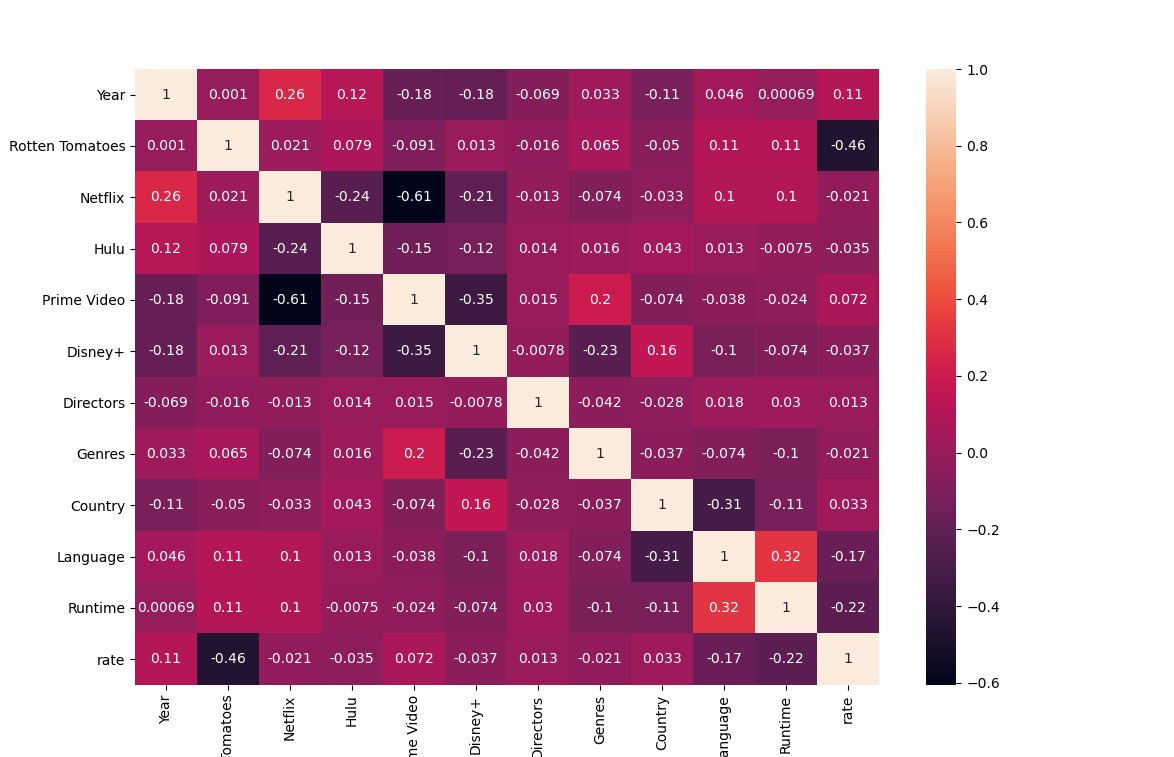
1. **Hyperparameter tuning:**
2. **SVM**
   * + 1. **Kernel**
       2. Linear (1 VS all) → it was the simplest and fastest kernel but didn’t deliver a good accuracy
       3. RBF (Radial Basis Function) → it is more complex than the Linear one and so it delivered a slightly more accurate result
       4. 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)
       5. **C (inverse of regularization parameter)**
3. Small (0.001) → the smaller it is the lower variance the model is and so delivered bad accuracy
4. Large (100) → the larger it is the higher variance the model is and so the test accuracy is bad
5. Moderate (1) → it is in the middle of the previous 2 and delivered a perfect balance so the model won’t overfit or underfit.
6. **Logistic Regression**
   * + 1. **C (inverse of regularization parameter)**
          1. Small (0.001) → the smaller it is the lower variance the model is and so delivered bad accuracy
          2. Large (100) → the larger it is the higher variance the model is and so the test accuracy is bad
          3. 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. **Solver/optimizer algorithm:**
7. ‘sag’ and ‘saga’ are faster for large data set. Saga was used because it supports elastic net (regression).
8. ‘liblinear’ is a good choice for small data set.
9. ‘lbfgs’, ‘sag’ and ‘saga’ handle L2.

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

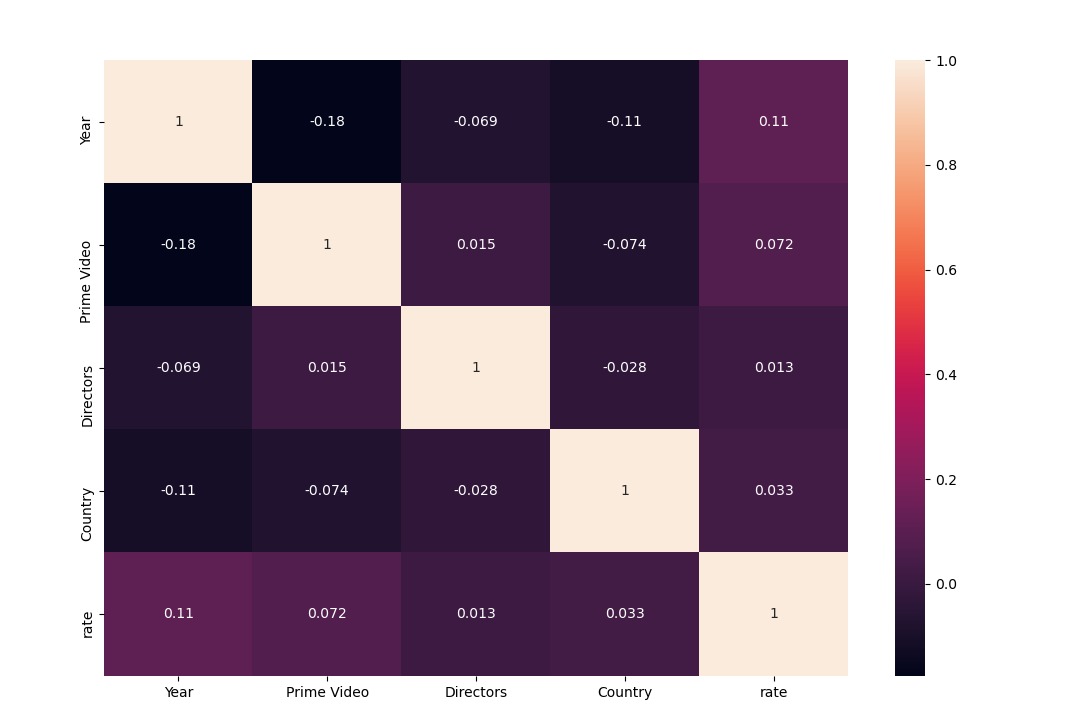
1. **Decision tree with Adaboost**
2. **No. of estimators (with default learning rate 1)**
3. Small (10) → it was the fastest and the accuracy wasn’t so bad.
4. Large (1000) → it was the slowest and overfits most of the time.
5. Moderate (100) → it delivered the best accuracy among the three but not the best among the 6.
6. **Learning rate**
7. Small (0.01) → is useful when using a large number of estimators or large depth (unlikely to cause overfitting)
8. 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)
9. Moderate (0.17) → it delivered the best accuracy

1. **Gradient Boosting**
2. **No. of estimators**
   * 1. Small (10) → it was the fastest and the accuracy wasn’t so bad
     2. Large (1000) →it was the slowest and overfits most of the time
     3. Moderate (100) → it delivered the best accuracy among the 6
3. **Learning rate**
   * 1. Small (0.01) → is useful when using a large number of estimators or large depth (unlikely to cause overfitting)
     2. 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)
     3. Moderate (0.17) → it delivered the best accuracy among the 3
4. **KNN:**
   1. **K-neighbors (3-9):**
5. Small → it was the fastest and the accuracy wasn’t so bad
6. Moderate → it delivered the best accuracy
7. Large →it was the slowest and overfits most of the time
   1. **Weight:**
8. Large k with distance →overfitting.
9. Small k with distance → accuracy wasn’t so bad
10. uniform → accuracy wasn’t so bad overfitting happens rarely.
11. **Conclusion:**
12. **Correlation between features:**

* **All features with rate:**

****

* **Features greater than 0:**

****

1. **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.

1. **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).

1. **Accuracy:**

**Training:**

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

**Testing:**

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