NETFLIX RATING PREDICTIONS

NESMA, ZIJING, AND YANNIS

PROJECT TIMELINE

1
Brainstorming and Idea Generation

2
Model Selection

3
Cleaning

4
EDA Tableau

5
Model Fitting

6
Model Testing

7
Results

CHALLENGES

CLEANING

Cleaning the cells and splitting when needed

WORKING SIMULTANEOUSLY

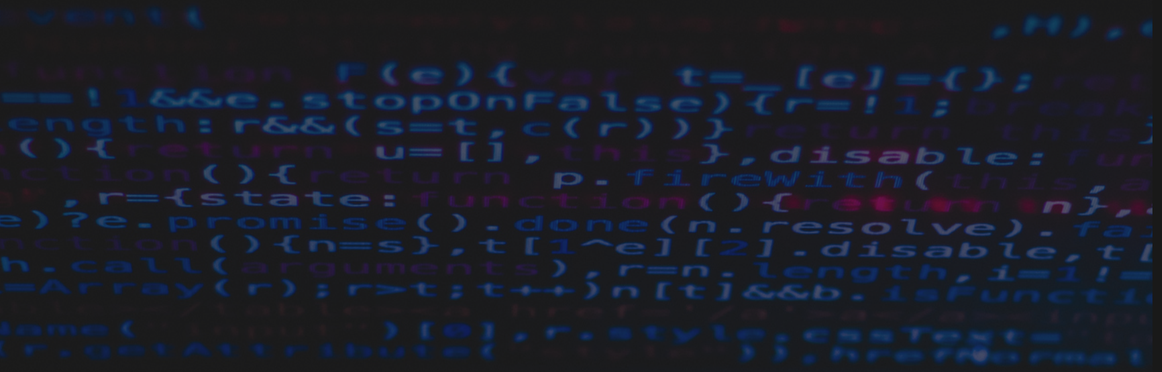
UNDERSTAND THE MODEL

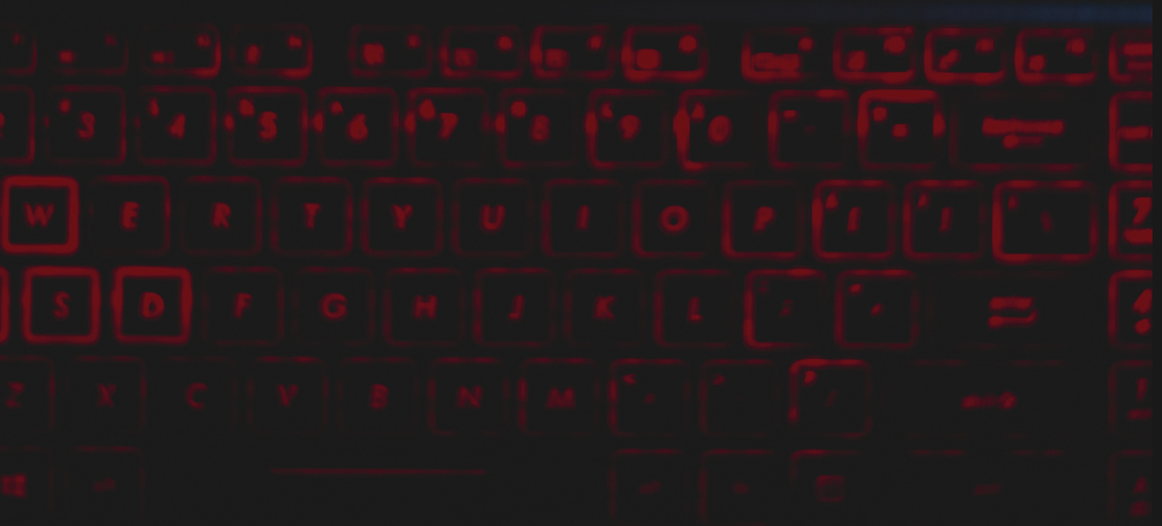
New models

INCREASE ACCURACY

Trying multiple ways to increase the accuracy of predictions

PREDICTION MODELS





1. SGD Classifier
2. Passive Aggressive Classifier
3. Random Forest Classifier

FEATURE SELECTION / SPLIT AND SCALE

```python
from sklearn.ensemble import RandomForestClassifier
from sklearn.feature_selection import SelectFromModel


X1 = netflix.drop(columns='rating')
y1 = round(netflix.rating)

# SelectFromModel: Meta-transformer for selecting features based on importance weights.
SFM = SelectFromModel(estimator=RandomForestClassifier(), max_features=4)
s=SFM.fit(X1, y1)
n_features = s.transform(X1).shape[1]
n_features
feature_idx = s.get_support()
feature_name = X1.columns[feature_idx]
feature_name
```

Python

```
Index(['year', 'vote', 'runtime', 'first_country'], dtype='object')
```

Split and scale

```python
from sklearn.model_selection import train_test_split
y = round(netflix.rating)
X = netflix[['year', 'vote', 'runtime', 'first_country']]
x_train, x_test, y_train, y_test= train_test_split(X, y, test_size=0.2)
```

Python

```python
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.transform(x_test)
```

Python

SGDCLASSIFIER

GridSearchCV

Looking for the best parameters

```
from sklearn.model_selection import GridSearchCV

params = {
    "loss" : ["hinge", "log", "squared_hinge", "modified_huber"],
    "alpha" : [0.0001, 0.001, 0.01, 0.1],
    "penalty" : ["l2", "l1", "none"],
}

model = SGDClassifier(max_iter=1000)
clf = GridSearchCV(model, param_grid=params)
```

```
clf.fit(x_train, y_train)
print(clf.best_score_)
print(clf.best_estimator_)
```

```
0.3165340143644674
SGDClassifier(alpha=0.1, loss='log')
```

Implement the best parameters in our model

```
sgdc = SGDClassifier(loss="log",max_iter=1000 , alpha=0.1)

sgdc.fit(x_train, y_train)

score = sgdc.score(x_train, y_train)
print("Training score: ", score)

cm = confusion_matrix(y_test, y_pred)
print(cm)

y_pred1 = sgdc.predict(x_test)
```

```
Training score:  0.3180700676090965
[[  0   0   0   0   0   5   4   4   0]
 [  0   0   0   0   5  13   5   5   0]
 [  0   0   0   0   8  46  13  13   0]
 [  0   0   0   0  10  72  44  17   0]
 [  0   0   0   3  34 172 104  59   0]
 [  0   0   0   7  41 237 152  82   0]
 [  0   0   0  11  19 217  94  79   0]
 [  0   0   0   1   4  31   6   9   0]
 [  0   0   0   0   0   0   0   1   0]]
```

PASSIVE AGGRESSIVE CLASSIFIER

PassiveAggressiveClassifier

```
from sklearn.linear_model import PassiveAggressiveClassifier
clf = PassiveAggressiveClassifier(max_iter=1000, random_state=10, tol=1e-3)
clf.fit(x_train, y_train)
y_pred = clf.predict(x_test)
#PassiveAggressiveClassifier(random_state=0)
print()
print(clf.coef_)
print(clf.intercept_)
print(y_pred)
```

Output exceeds the size limit. Open the full output data in a text editor

```
generate_results(clf, y_pred, 'PassiveAggressiveClassifier')
```

The classification report for PassiveAggressiveClassifier is:

	precision	recall	f1-score	support
1	0.00	0.00	0.00	0
2	0.00	0.00	0.00	5
3	0.00	0.00	0.00	14
4	0.00	0.00	0.00	54
5	0.00	0.00	0.00	104
6	0.33	0.88	0.48	260
7	0.33	0.05	0.08	255
8	0.31	0.07	0.12	107
9	0.00	0.00	0.00	15
accuracy			0.31	814
macro avg	0.11	0.11	0.08	814
weighted avg	0.25	0.31	0.20	814

RANDOM FOREST CLASSIFIER

```
from sklearn.model_selection import GridSearchCV, RandomizedSearchCV
rfc = RandomForestClassifier()
rfc.fit(x_train, y_train)

parameter_grid = {'n_estimators':np.arange(5,200,5), 'max_features':["auto", "sqrt", "log2"], 'criterion':["gini", "entropy"]}

#scoring: the goal, to fing the optimal no. of hyperparam n_estimators(no. of tree) for RandomForestClassifier model
grid_search = GridSearchCV(RandomForestClassifier(random_state=0),
                           parameter_grid,cv=5, scoring="accuracy",verbose=1,n_jobs=-1
                           )
grid_search.fit(x_train,y_train)
```

```
Fitting 5 folds for each of 234 candidates, totalling 1170 fits

GridSearchCV(cv=5, estimator=RandomForestClassifier(random_state=0), n_jobs=-1,
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_features': ['auto', 'sqrt', 'log2'],
                         'n_estimators': array([  5,  10,  15,  20,  25,  30,  35,  40,  45,  50,  55,  60,  65,
        70,  75,  80,  85,  90,  95, 100, 105, 110, 115, 120, 125, 130,
       135, 140, 145, 150, 155, 160, 165, 170, 175, 180, 185, 190, 195])},
             scoring='accuracy', verbose=1)
```

```
grid_search.best_params_
```

```
{'criterion': 'gini', 'max_features': 'auto', 'n_estimators': 195}
```

```
grid_search.best_score_
```

```
0.4967751381130667
```

RANDOM FOREST CLASSIFIER / SECOND SPLIT

```
x_train09, x_test09, y_train09, y_test09 = train_test_split(X, y, test_size=0.1, random_state=5)
```

```
rfc3 = RandomForestClassifier(criterion= 'gini', max_features='log2', n_estimators=190)
model3 = rfc3.fit(x_train09, y_train09)
y_pred09 = model09.predict(x_test09)
sklearn.metrics.r2_score(y_test09, y_pred09)
y_pred09 = model3.predict(x_test09)
y_pred09 = rfc3.predict(x_test09)
acc_rfc3 = rfc3.score(x_test09, y_test09)
print('The accuracy of the Random Forest Classifier is:', acc_rfc3 * 100, '%')
```

```
The accuracy of the Random Forest Classifier is: 52.08845208845209 %
```

RANDOM FOREST CLASSIFIER / SECOND SPLIT

Training on the full data and evaluation on test data

```
rfc3 = RandomForestClassifier(criterion= 'gini', max_features='auto', n_estimators=195)
model3 = rfc3.fit(X, y)
```

```
y_test_pred2 = model3.predict(x_test)
sklearn.metrics.r2_score(y_test, y_test_pred2)
```

```
1.0
```

```
acc_rfc3 = rfc3.score(x_test, y_test)
print('The accuracy of the Random Forest Classifier is:', acc_rfc3 * 100, '%')
generate_results(rfc3, y_pred, 'Random Forest Classifier')
```

```
The accuracy of the Random Forest Classifier is: 100.0 %
```

NETFLIX

QUESTIONS?