TMDB Revenue Prediction

FITE3010 Big Data and Data Mining Project

LAU Yan Chun Chris 3035790941

INDEX

#1: Aim and Objective

data_type ✖

```
1 print(data.shape)
2 print(data.iloc[0])
```

```
(3000, 23)
id                                                                1
belongs_to_collection    [{'id': 313576, 'name': 'Hot Tub Time Machine ...
budget                                                     14000000
genres                                  [{'id': 35, 'name': 'Comedy'}]
homepage                                                         NaN
imdb_id                                                    tt2637294
original_language                                                 en
original_title                                   Hot Tub Time Machine 2
overview                 When Lou, who has become the "father of the In...
popularity                                                  6.575393
poster_path                                /tQtWuwvMf0hCc2QR2tkolwl7c3c.jpg
production_companies     [{'name': 'Paramount Pictures', 'id': 4}, {'na...
production_countries     [{'iso_3166_1': 'US', 'name': 'United States o...
release_date                                                  2/20/15
runtime                                                          93.0
spoken_languages                [{'iso_639_1': 'en', 'name': 'English'}]
status                                                       Released
tagline                  The Laws of Space and Time are About to be Vio...
title                                            Hot Tub Time Machine 2
Keywords                 [{'id': 4379, 'name': 'time travel'}, {'id': 9...
cast                     [{'cast_id': 4, 'character': 'Lou', 'credit_id...
crew                     [{'credit_id': '59ac067c92514107af02c8c8', 'de...
revenue                                                     12314651
Name: 0, dtype: object
```

1. Given data, predict the revenue of movie(s)
2. Standard Machine Learning/Statistics regression task.

   Regression(Data) -> Value

1. **22** features in total obtained from DB.

X(22)

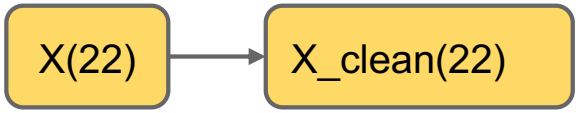
#2
Data Cleaning

Check NaN value for DataFrame:	data_train	data_test
id	0	0
belongs_to_collection	2396	3521
budget	0	0
genres	7	16
homepage	2054	2978
imdb_id	0	0
original_language	0	0
original_title	0	0
overview	8	14
popularity	0	0
poster_path	1	1
production_companies	156	258
production_countries	55	102
release_date	0	1
runtime	2	4
spoken_languages	20	42
status	0	2
tagline	597	863
title	0	3
Keywords	276	393
cast	13	13

Question: Which feature(s) should be cleaned?

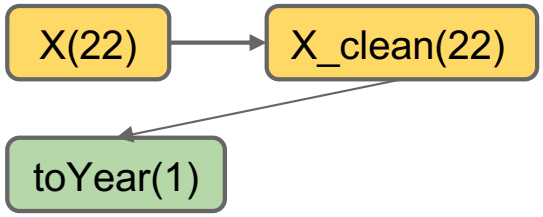
By checking *NaN* value in DB:

1. Release_date -> mode
2. Runtime -> mode
3. Title -> ''
4. tagline -> ''
5. Overview -> ''

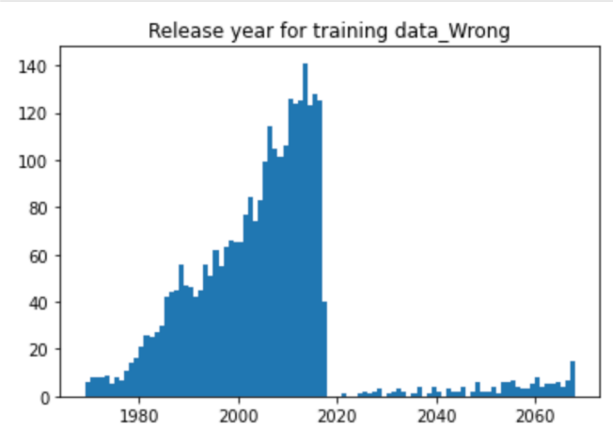


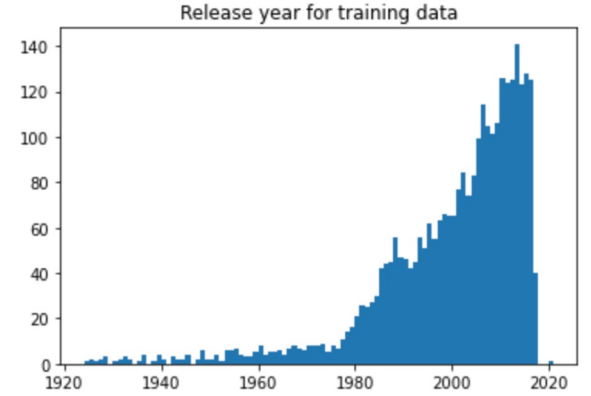
#3 Data Preprocessing(1) “release_date”

#1 Target: Time data: release_date




```
def toYear(col):
    res = []
    for xi in col:
        time_object = datetime.strptime(xi, '%m/%d/%y').date()
        time_year = int(time_object.year)
#         if (time_year >= 2023):
#             time_year -= 100
        res.append(time_year)
    return res
```

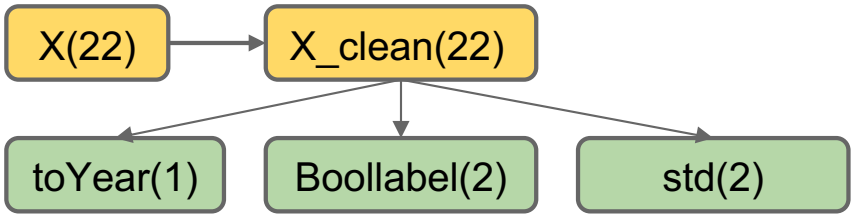




#3 Data Preprocessing(2)
Boolean Label and Float data

#2 Target: string data -> boolean label

#3 Target: float data: budget, popularity




```
# boolean label :0 or 1
hmpage_label = generateBool(data["homepage"])
poster_label = generateBool(data["poster_path"])
```

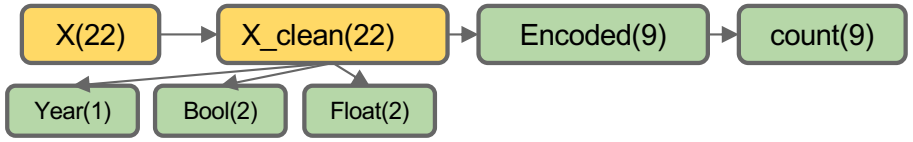
```
#preprocessing for float data

num_scaler = StandardScaler()

data_num_std = [data['budget'],data['popularity']]
df_num_std = pd.DataFrame(data_num_std).T

df_num_std = num_scaler.fit_transform(df_num_std)
df_num_std = pd.DataFrame(df_num_std)
df_num_std.columns = ["budget_std","popularity_std"]
```

#3 Data Preprocessing(3.1) Encoded dict() string

#4 Target: String data in dict() format

1. Extract the id/name from string in form of dictionary

1. 3 Products in total
   a. Feature_list: str [ ]
   b. Feature_label: str [ ] [ ]
   c. Feature_count: int [ ]



Example of genres:

```
1 genres_count[0:5]
```

```
[1, 4, 1, 2, 2]
```

```
1 genres_list[0:5]
```

```
['10402', '10749', '10751', '10752', '10769']
```

```
1 genres_label[0:5]
```

```
[['35'], ['35', '18', '10751', '10749'], ['18'], ['53', '18'], ['28', '53']]
```

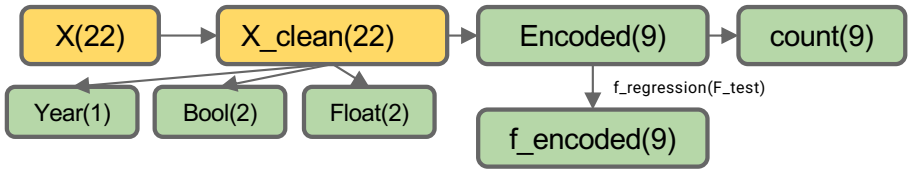
```
# for training data
# extract clean data from string in the form of dictionary
coll_label, coll_list = generateTotalList(data["belongs_to_collection"], "{'id': ",",")
genres_label, genres_list = generateTotalList(data["genres"],"'id': ",",")
prodComp_label, prodComp_list = generateTotalList(data["production_companies"],"'id': ",}")
prodCtry_label, prodCtry_list = generateTotalList(data["production_countries"],"'iso_3166_1': '","'")
original_label, original_list = generateTotalList(data["original_language"],None,None)
spoken_label, spoken_list  = generateTotalList(data["spoken_languages"],"'iso_639_1': '","'")
keyword_label, keyword_list = generateTotalList(data["Keywords"],"{'id': ",",")
cast_label, cast_list = generateTotalList(data["cast"],", 'id': ",",")
crew_label, crew_list = generateTotalList(data["crew"],", 'id': ",",")
```

#3 Data Preprocessing(3.2) Encoded (cont'd)

#5 Target: "key" features in encoded features

Total No. production companies: **3712**

Question: Which are suitable to be considered?
Answer: Filter encoded features using **F-regression**.




```python
def fRegressionTest_encoded(encoded_list,name):
    f, p = f_regression(X = encoded_list, y = data['revenue'])
    print("--",name,"--")
    print("original No. features:","\t",len(p),"\tfeatures selected:", sum(p < 0.05))
    res_bool = p < 0.05
    res = encoded_list[encoded_list.columns[res_bool]]
    return res
```

```
-- collection --
original No. features:   422    features selected: 59
-- genres --
original No. features:   20     features selected: 11
-- production companies --
original No. features:   3712   features selected: 172
-- production countries --
original No. features:   74     features selected: 8
-- original language --
original No. features:   36     features selected: 5
-- spoken languages --
original No. features:   79     features selected: 7
-- keywords --
original No. features:   7400   features selected: 560
-- cast --
original No. features:   38760  features selected: 3273
-- crew --
original No. features:   38897  features selected: 4277
```

#4 Model Selection

```
regr = RandomForestRegressor()
param = {'n_estimators': [50,100,150,200,250,500],
        'criterion': ['squared_error', 'friedman_mse'],
        'max_depth': [10,25,50,None]}

gs = GridSearchCV(regr, param, cv = 5, n_jobs = -1,
                  scoring = 'neg_root_mean_squared_error',
                  verbose = 1).fit(X_train, y_train)

```

```
Fitting 5 folds for each of 48 candidates, totalling 240 fits
```

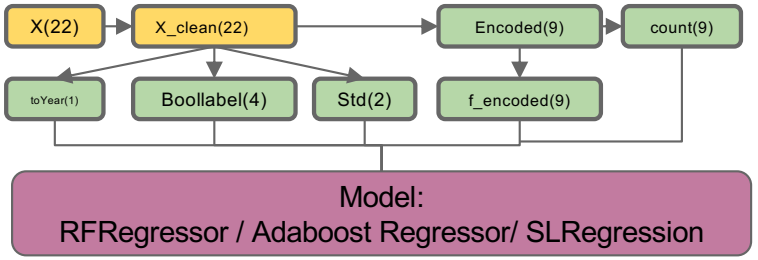
```
gs.best_params_
```

```
{'criterion': 'friedman_mse', 'max_depth': 10, 'n_estimators': 200}
```

```
slr = LinearRegression().fit(X_train,y_train)
ada = AdaBoostRegressor().fit(X_train, y_train)
```

Three Models are selected:

1. RandomForestRegressor()
   (Using GridSearchCV for tuning)
2. LinearRegression()
3. AdaboostRegressor()

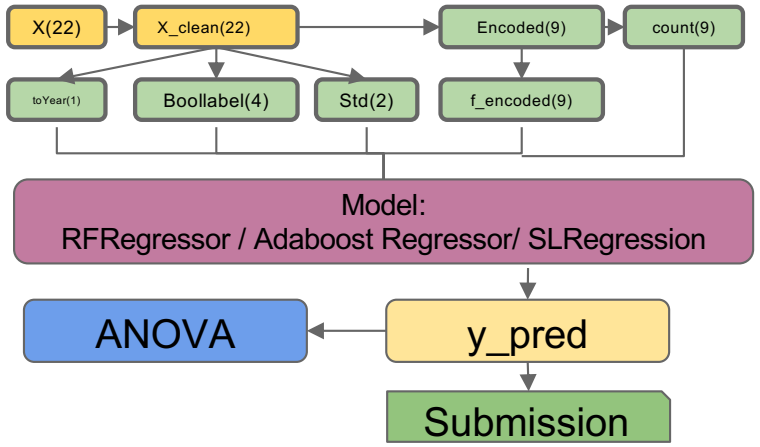


#5 Performance Analysis - ANOVA

```python
y_valid_pred_regr = gs.predict(X_valid)
y_valid_pred_slr = slr.predict(X_valid)
y_valid_pred_ada = ada.predict(X_valid)

def anovaFtest(y, y_pred,n_r, name_model):
    mean_y = statistics.mean(y)
    mean_y_pred = statistics.mean(y_pred)
    ssr = 0
    sse = 0
    n_t = len(y)-1
    n_e = n_t - n_r
    for yi, yi_pred in zip(y,y_pred):
        ssr += (yi_pred - mean_y)**2
        sse += (yi - yi_pred)**2
    msr = ssr/n_r
    mse = sse/n_e
    f_ratio = msr/mse
    f_critical = scipy.stats.f.ppf(q = 1- 0.05, dfn = n_r, dfd = n_t)
    print("---- ANOVA for",name_model,"----")
    print("MSR:",msr,'\t',"MSE:",mse)
    print("F-critical:", f_critical, '\t',"degree of Freedom:", n_r, n_t)
    print("F-value:",f_ratio)

anovaFtest(y_valid,y_valid_pred_regr, X_train.shape[1], "Random Forest Regressor")
anovaFtest(y_valid,y_valid_pred_slr, X_train.shape[1], "Linear Regression")
anovaFtest(y_valid,y_valid_pred_ada, X_train.shape[1], "Adaboost Regressor")
```

```
---- ANOVA for Random Forest Regressor ----
MSR: 1.0939018004605838e+17      MSE: 6006424089023688.0
F-critical: 1.2903958928993684   degree of Freedom: 85 599
F-value: 18.212197211642305
---- ANOVA for Linear Regression ----
MSR: 8.242710374058611e+16       MSE: 1.0325165772340464e+16
F-critical: 1.2903958928993684   degree of Freedom: 85 599
F-value: 7.983126427025093
---- ANOVA for Adaboost Regressor ----
MSR: 1.2488383600938632e+17      MSE: 1.3598520274411592e+16
F-critical: 1.2903958928993684   degree of Freedom: 85 599
F-value: 9.183634210876672
```

For the analysis, **ANOVA** F-test is applied to determine whether the predicted results are statistically significant or not.



ANOVA in one sentence:
How statistically significant is the model in explaining the variance of the result(y) with its prediction(y_pred).

#6 Submission

1. 22 features + 1 predictor(revenue) are obtained
2. **85** Features in total for training
3. RandomForestRegressor model is applied
4. GridSearchCV is applied for tuning
5. Prediction submitted on Kaggle with score: **2.65727**

Submission and Description	Private Score	Public Score
submission_3035790941_4.csv Complete (after deadline) · 15m ago	**2.65727**	**2.65727**

#7 Discussion(1) - Word Analysis

Several attempts to further extract “keywords” in the content from the raw data as follows:

Target:
(a)overview, (b)tagline, (c)title

Result: **Good** (Score: 2.19~2.21 on Kaggle)

```python
from sklearn.feature_extraction.text import CountVectorizer
import nltk
from nltk.corpus import stopwords
stop_words = list(stopwords.words('english'))

overview_cv = CountVectorizer(stop_words = stop_words)
tagline_cv = CountVectorizer(stop_words = stop_words)
title_cv = CountVectorizer(stop_words = stop_words)

#train
overview_vectorized = overview_cv.fit_transform(data["overview"])
tagline_vectorized = tagline_cv.fit_transform(data["tagline"])
title_vectorized = title_cv.fit_transform(data["title"])

df_overview = pd.DataFrame(overview_vectorized.toarray(),columns = overview_cv.get_feature_names_out())
df_tagline = pd.DataFrame(tagline_vectorized.toarray(), columns = tagline_cv.get_feature_names_out())
df_title = pd.DataFrame(title_vectorized.toarray(), columns = title_cv.get_feature_names_out())
```

```
-- collection --
original No. features:   422    features selected: 28
-- genres --
original No. features:   20     features selected: 12
-- production companies --
original No. features:   3712   features selected: 285
-- production countries --
original No. features:   74     features selected: 16
-- original language --
original No. features:   36     features selected: 12
-- spoken languages --
original No. features:   79     features selected: 8
-- keywords --
original No. features:   7400   features selected: 325
-- cast --
original No. features:   38760  features selected: 1529
-- crew --
original No. features:   38897  features selected: 2336
-- overview --
original No. features:   17301  features selected: 1028
-- tagline --
original No. features:   3184   features selected: 114
-- title --
original No. features:   3297   features selected: 183
```

#7 Discussion(2) - Word Analysis - limited. No

Limit to max. No. features selected from the content, Saying 30 encoded labels for each feature.

```
-- collection --
original No. features:    422    features selected: 28
-- genres --
original No. features:    20     features selected: 12
-- production companies --
original No. features:    3712   features selected: 285
-- production countries --
original No. features:    74     features selected: 16
-- original language --
original No. features:    36     features selected: 12
-- spoken languages --
original No. features:    79     features selected: 8
-- keywords --
original No. features:    7400   features selected: 325
-- cast --
original No. features:    38760  features selected: 1529
-- crew --
original No. features:    38897  features selected: 2336
-- overview --
original No. features:    17301  features selected: 1028
-- tagline --
original No. features:    3184   features selected: 114
-- title --
original No. features:    3297   features selected: 183
```

```python
def fRegressionTest_encoded(encoded_list,name):
    f, p = f_regression(X = encoded_list, y = data['revenue'])
    index = 1
    res_bool = (p < 0.05*index)
    print("--",name,"--")
# #     decrease the p value to further reduce the number of features going to be selected
#     while(sum(res_bool) >= 70):
#         print("original No. features:","\t",len(p),"\tfeatures selected:", sum(res_bool))
#         index *= 0.8
#         res_bool = p < 0.05*index

    print("original No. features:","\t",len(p),"\tfeatures selected:", sum(res_bool))
    res = encoded_list[encoded_list.columns[res_bool]]
    return res
```

#8: Q&A

Thank You!