Fouille de données - Kaggle Project Final Report

Raymond Klutse
Ricardo Rodriguez
Qu Runlu
Rudresh Mishra

Date: 1 avril 2020

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

TABLE OF CONTENTS

1. Introduction

This project is coming from Kaggle TMDB Box Office Prediction Challenge.The film industry has been booming over the year and has thus created massive data about films. A lot more companies are making films with the aim of earning money ,thereby increasing their revenue. In this project, we explore the use of machine learning to predict film revenues.

2. Context and Problem Understanding

In a world, where movies made an estimated $41.7 billion in 2018, the film industry is more popular than ever. But what movies make the most money at the box office? How much does a director matter? Or the budget? For some movies, it's "You had me at 'Hello.'" For others, the trailer falls short of expectations and you think "What we have here is a failure to communicate."

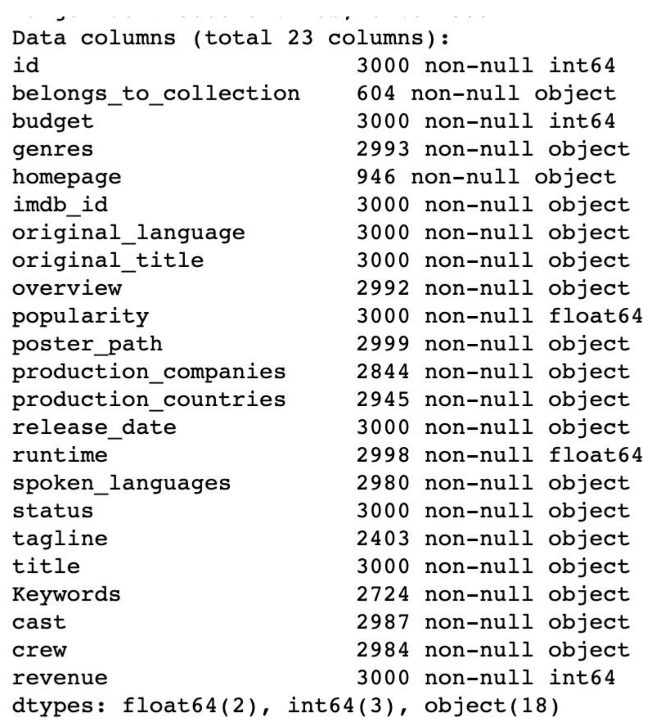
In this competition, we are presented with metadata on over 7,000 past films from The Movie Database to try and predict their overall worldwide box office revenue. Data points provided include cast, crew, plot keywords, budget, posters, release dates, languages, production companies, and countries. You can collect other publicly available data to use in your model predictions, but in the spirit of this competition, use only data that would have been available before a movie's release.

3. Data Understanding and Exploration , Feature Engineering

3.1 Data Understanding

We have 7398 movies and a variety of metadata obtained from The Movie Database (TMDB). The data set has been divided into two parts, the train dataset (3000 movies) and test dataset(4398 movies).

Movies are labeled with id. Data points include ‘belongs_to_collection’, ‘budget’, ‘genres’, ‘homepage’, ‘imdb_id’, ‘original_language’, ‘overview’, ‘popularity’, ‘poster_path’, ‘production companies’, ‘production_countries’, ‘release_date’, ‘runtime’, ‘spoken_language’, ‘status’, ‘tagline’, ‘title’, ‘Keywords’, ‘cast’, ‘crew’, ‘revenue’ (the test data set does not contain this column).



```
Data columns (total 23 columns):
id                       3000 non-null int64
belongs_to_collection    604 non-null object
budget                   3000 non-null int64
genres                   2993 non-null object
homepage                 946 non-null object
imdb_id                  3000 non-null object
original_language        3000 non-null object
original_title           3000 non-null object
overview                 2992 non-null object
popularity               3000 non-null float64
poster_path              2999 non-null object
production_companies     2844 non-null object
production_countries     2945 non-null object
release_date             3000 non-null object
runtime                  2998 non-null float64
spoken_languages         2980 non-null object
status                   3000 non-null object
tagline                  2403 non-null object
title                    3000 non-null object
Keywords                 2724 non-null object
cast                     2987 non-null object
crew                     2984 non-null object
revenue                  3000 non-null int64
dtypes: float64(2), int64(3), object(18)
```

Fig 1: the features in the original data set

Among these features, there are six columns storing their information as a dictionary, which means we have to clean the data to extract more useful information.

3.2 Data Preprocessing

After having a slight look at the data source, we noticed that there exist a lot of missing data and some unanalysable columns, let's start cleaning data and dealing with missing data.

3.2.1 Missing data

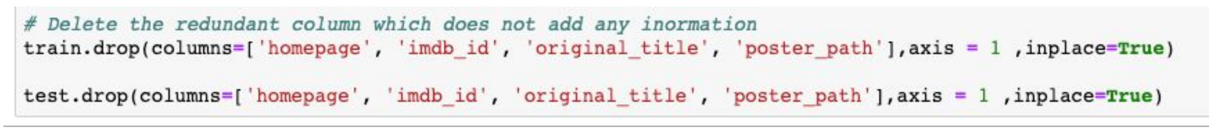
As all the missing values are ‘string’ or ‘dict’, we replaced them by empty values which would not make huge differences at the moment of encoding.

We notice that for the budget value there were some zero values. Since the quantity of these rows was almost a quarter of the dataset, we decided not to drop it but to replace it by the mean value.

3.2.2 Cleaning data

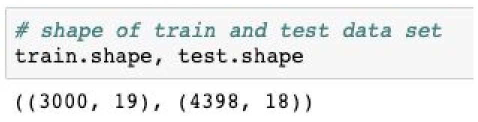
We took out the information from the column which has the type as dictionary .(such as: ‘belong_to_collection’, ‘crew’)

After look at all the features, we dropped 4 features that we would not use.(‘homepage’, ‘imdb_id’, ‘original_title’, ‘poster_path’)



```
# Delete the redundant column which does not add any inormation
train.drop(columns=['homepage', 'imdb_id', 'original_title', 'poster_path'],axis = 1 ,inplace=True)

test.drop(columns=['homepage', 'imdb_id', 'original_title', 'poster_path'],axis = 1 ,inplace=True)
```

Fig 2: dropped columns



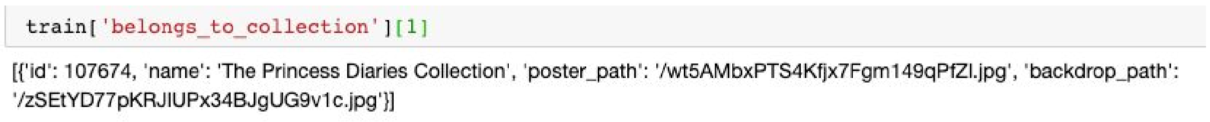
```
# shape of train and test data set
train.shape, test.shape
```

```
((3000, 19), (4398, 18))
```

Fig 3: the training set size and the testing set size

Extract information from the columns which are the type of dictionary 'belongs_to collection'

- 'belongs_to_collection'



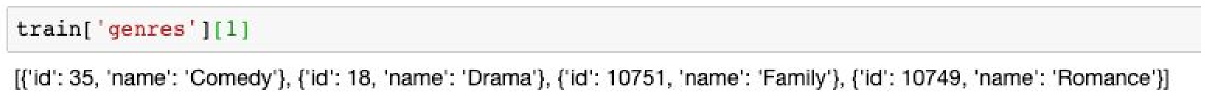
```
train['belongs_to_collection'][1]
```

```
[{'id': 107674, 'name': 'The Princess Diaries Collection', 'poster_path': '/wt5AMbxPTS4Kfjx7Fgm149qPfZl.jpg', 'backdrop_path':
'/zSEtYD77pKRJlUPx34BJgUG9v1c.jpg'}]
```

Fig 4: an example of 'belongs_of_collection'

This feature contains information about the film's collection. If the film belongs to a collection, then it will have the collection_id, conllection_name. But those are not so important, what we are concerned about is if this film belongs to a collection, so we create a new column 'has_collection' which is binary type (0 and 1). If it belongs to a collection, we sign it with '1', otherwise '0'.

- 'genre'



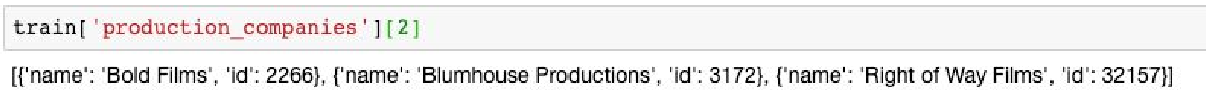
```
train['genres'][1]
```

```
[{'id': 35, 'name': 'Comedy'}, {'id': 18, 'name': 'Drama'}, {'id': 10751, 'name': 'Family'}, {'id': 10749, 'name': 'Romance'}]
```

Fig 5: an example of 'genre'

'genre' represents the type of film, the film could be considered as different types. Here is an example of 'genre'. So based on that, we suppose that the amount of types will influence the revenue since it greets different groups of audiences.At the same time, we store the types as a list for each film. For further analysis, the amount of types should be normalized to measure its importance. We calculate the weight of each type (such as: 'roman', 'comedy'), since we stored the type of film into a list, picking the maximum weight in the list to normalize the importance.
Three new columns have been created related to 'genre': 'Top_genre', 'All_genre', 'max_genre_norm'.

- 'production_companies'



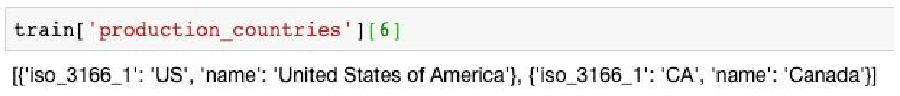
```
train['production_companies'][2]
```

```
[{'name': 'Bold Films', 'id': 2266}, {'name': 'Blumhouse Productions', 'id': 3172}, {'name': 'Right of Way Films', 'id': 32157}]
```

Fig 6: an example of 'production_companies'

‘production_companies’ is the column which contains the film’s production companies’ information. A film could be made by several production companies, so the same as ‘genre’, we created three new columns: ‘num_companies’, ‘all_production_companies’, ‘max_production_companies_norm’.

- ‘production_countries’



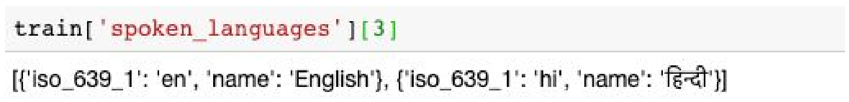
```
train['production_countries'][6]
```

```
[{'iso_3166_1': 'US', 'name': 'United States of America'}, {'iso_3166_1': 'CA', 'name': 'Canada'}]
```

Similar to previous variables, the amount of production countries and which country it is would be very interesting when we forecast the revenue.

Three new columns: ‘num_production_countries’, ‘max_production_country_norm’, ‘all_production_countries’

- ‘spoken_language’



```
train['spoken_languages'][3]
```

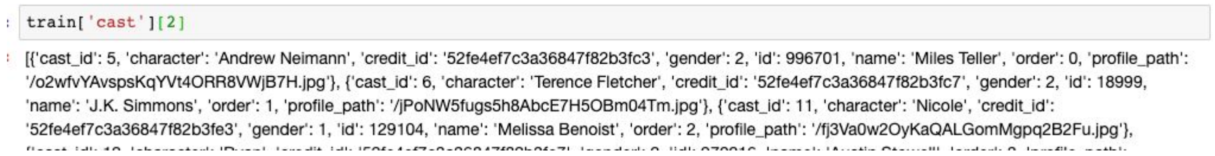
```
[{'iso_639_1': 'en', 'name': 'English'}, {'iso_639_1': 'hi', 'name': 'हिन्दी'}]
```

Fig 7: an example of ‘spoken_language’

It is commonly an important feature when we talk about the film. If using a more universal language and more language version it provides, more population could be exposed to it.

Three new columns created: ‘num_languages’, ‘all_languages’, ‘max_language_norm’.

- ‘cast’



```
train['cast'][2]
```

```
[{'cast_id': 5, 'character': 'Andrew Neimann', 'credit_id': '52fe4ef7c3a36847f82b3fc3', 'gender': 2, 'id': 996701, 'name': 'Miles Teller', 'order': 0, 'profile_path':
'/o2wfvYAvspsKqYVt4ORR8VWjB7H.jpg'}, {'cast_id': 6, 'character': 'Terence Fletcher', 'credit_id': '52fe4ef7c3a36847f82b3fc7', 'gender': 2, 'id': 18999,
'name': 'J.K. Simmons', 'order': 1, 'profile_path': '/jPoNW5fugs5h8AbcE7H5OBm04Tm.jpg'}, {'cast_id': 11, 'character': 'Nicole', 'credit_id':
'52fe4ef7c3a36847f82b3fe3', 'gender': 1, 'id': 129104, 'name': 'Melissa Benoist', 'order': 2, 'profile_path': '/fj3Va0w2OyKaQALGomMgpq2B2Fu.jpg'},
```

Fig 8: an example of ‘cast’

It contains all the casts’ name and character information. Since a film has several casts, maybe some casts can lead a high revenue to the film. Based on the cast's name, we take three columns to further analyse: ‘num_cast’, ‘all_cast’, ‘max_cast_norm’.
We think the cast's gender could also be interesting, for exploring the gender, we count the amount of cast with different genders. ‘genders_0_cast’ is the unknown gender, ‘genders_1_cast’ is the amount of females in the film while ‘‘genders_2_cast’’ represent the amount of males in the film.

- ‘crew’

‘crew’ represents the staff’s information of a film.

10 new columns are prepared for exploring: ‘num_crew’, ‘crew_director’, ‘crew_director_norm’, ‘crew_director_log_norm’, ‘crew_producer’, ‘crew_producer_norm’, ‘crew_producer_log_norm’, ‘crew_composer_norm’, ‘crew_composer_log_norm’, ‘crew_composer’.

- ‘released_date’

Thinking about the released date makes a difference for the revenue. Maybe during the summer, a lot of people are on vacation then they have time to see the movies.
so we take ‘release_year’, ‘release_year’, ‘release_month’, ‘day_of_week’ for further exploration.

3.3 Data Exploration

1. ‘has_collection’ / ‘has_homepage’/ ‘has_tagline’/ ‘has_released with revenue

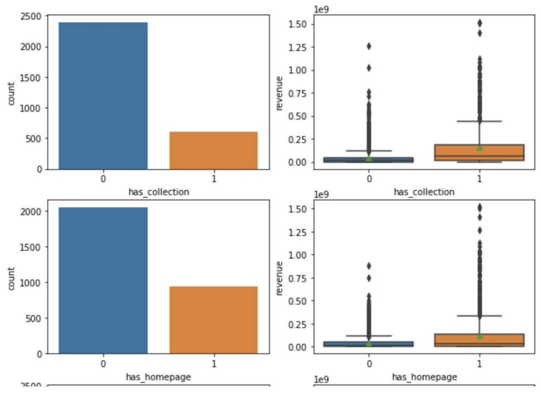


Fig 9: ‘has_collect’ and ‘has homepage’

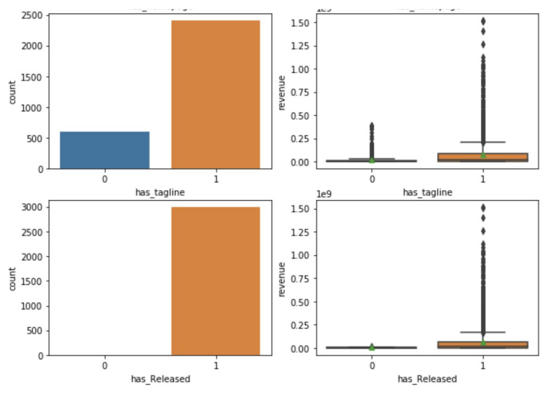


Fig 10: ‘has_tagline’ and ‘has_released’

From the above chart, we can observe the following information:

- The number of the collection movies is less than that of non-collection movies, but the box office average and median of collection movies are higher than non-collection movies.(Fig 9)

- Movies with an official homepage are fewer than movies without a homepage, but the average box office and median of movies with a homepage are higher than movies without a homepage. (Fig 9)

- Most films have a tagline. Similarly, films with tagline have higher box office averages and median than films without tagline. (Fig 10)

- Only a few movies are non-released, and the box office average and median of released movies are higher than those of non-released movies. (Fig 10)

2. ‘budget’/‘genre’ count/‘spoken_language’/ with ‘revenue’

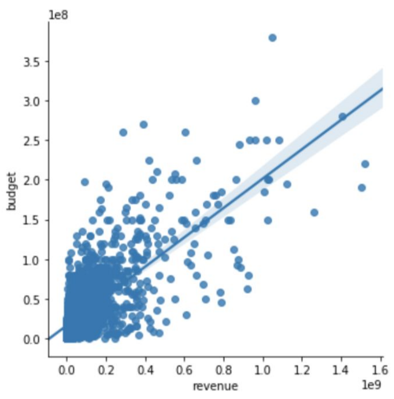
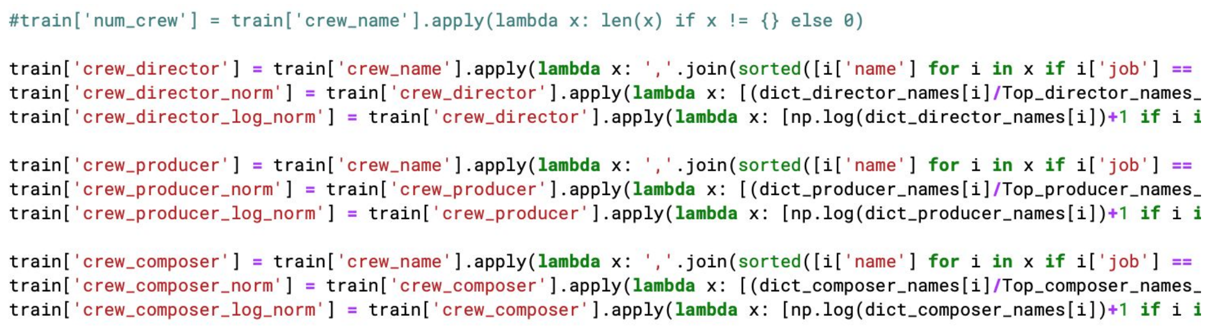


Fig 11: ‘budget’ with ‘revenue’

From this figure, we could see that ‘budget’ is positively correlated to ‘revenue’. With ‘budget’ increasing, the ‘revenue’ is increasing. (Fig 11)

3.4 Additional Feature Engineering

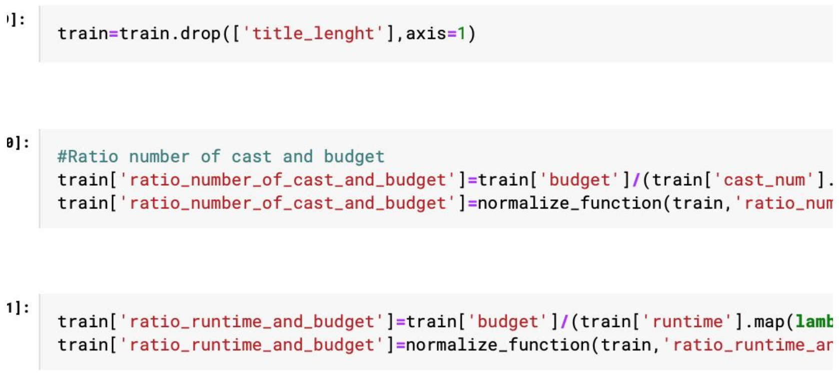
We created new features in order to explore more relation between the different columns.We created three features namely crew_director_log_norm, crew_producer_log_norm, rew_composer_log_norm. They reflect the log of number of times director,producer and composer has worked in the movies.(Fig 12)



```
#train['num_crew'] = train['crew_name'].apply(lambda x: len(x) if x != {} else 0)

train['crew_director'] = train['crew_name'].apply(lambda x: ','.join(sorted([i['name'] for i in x if i['job'] ==
train['crew_director_norm'] = train['crew_director'].apply(lambda x: [(dict_director_names[i]/Top_director_names_
train['crew_director_log_norm'] = train['crew_director'].apply(lambda x: [np.log(dict_director_names[i])+1 if i i

train['crew_producer'] = train['crew_name'].apply(lambda x: ','.join(sorted([i['name'] for i in x if i['job'] ==
train['crew_producer_norm'] = train['crew_producer'].apply(lambda x: [(dict_producer_names[i]/Top_producer_names_
train['crew_producer_log_norm'] = train['crew_producer'].apply(lambda x: [np.log(dict_producer_names[i])+1 if i i

train['crew_composer'] = train['crew_name'].apply(lambda x: ','.join(sorted([i['name'] for i in x if i['job'] ==
train['crew_composer_norm'] = train['crew_composer'].apply(lambda x: [(dict_composer_names[i]/Top_composer_names_
train['crew_composer_log_norm'] = train['crew_composer'].apply(lambda x: [np.log(dict_composer_names[i])+1 if i i
```

Fig 12

The second new feature is the ratio between number of cast and budget which is normalization of division of budget to number of cast and the third feature is normalization of division to budget/ runtime as shown below: (Fig 13)



```
]:   train=train.drop(['title_lenght'],axis=1)


0]:  #Ratio number of cast and budget
     train['ratio_number_of_cast_and_budget']=train['budget']/(train['cast_num'].
     train['ratio_number_of_cast_and_budget']=normalize_function(train,'ratio_num


1]:  train['ratio_runtime_and_budget']=train['budget']/(train['runtime'].map(lamb
     train['ratio_runtime_and_budget']=normalize_function(train,'ratio_runtime_ar
```

Fig 13

4. Feature Selection

In order to avoid overfitting our model,reduce training time and make it more explainable, we applied some feature selection methods in order to select important features for training. The methods used and their respective results are as follows:

4.1 Filter Methods

- Pearson Correlation

We used this method to select 87 features that are important in the dataset.

4.2 Wrapper Methods

- Forward Selection

We used this method to select 87 features that are important in the dataset

4.3 Embedded methods

- Random Forest Regressor

This method returned 87 features that were deemed important for training the model.

5. Modeling

Our problem is a regression problem to predict the revenue of a movie. For this reason, we selected 3 machine learning models that would help us solve the problem.

The training set consists of 2698 individuals and 185 features whilst the validation set consists of 300 individuals and 185.

We trained 3 machine learning models using the different features selected by each of the feature selection methods used above. The models used were Linear Regression, Decision Tree, Random Forest.

5.1 Linear Regression

A linear regression model was trained with the different features selected by each of the feature selection methods. All Linear Regression parameters were left in their default state.

5.2 Decision Tree

A decision tree model was trained using the different features selected by each of the feature selection methods. The Decision Tree Regressor was trained using different sizes of tree depths, ranging from 2 to 100. The random state of the decision tree regressor was set to 0.

5.3 Random Forest Regressor

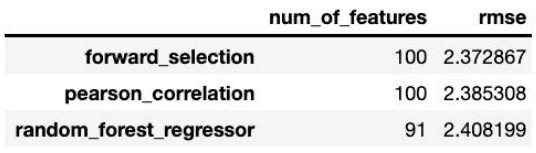
A random forest regressor model was trained using the different features selected by each of the feature selection methods. The Random forest regressor was trained using different sizes of tree depths, ranging from 20 to 1500. The random state of the decision tree regressor was set to 0.

6. Evaluation

6.1 Normal Evaluation

We show here the results of each of the models after training.

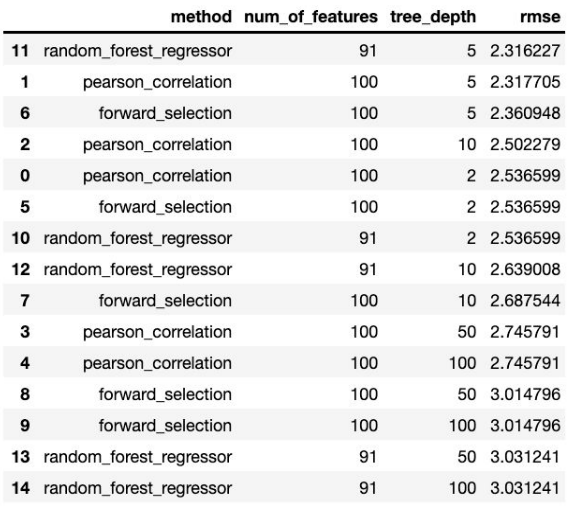
- Linear Regression



	num_of_features	rmse
forward_selection	100	2.372867
pearson_correlation	100	2.385308
random_forest_regressor	91	2.408199

Fig 14: the result of applying linear regression

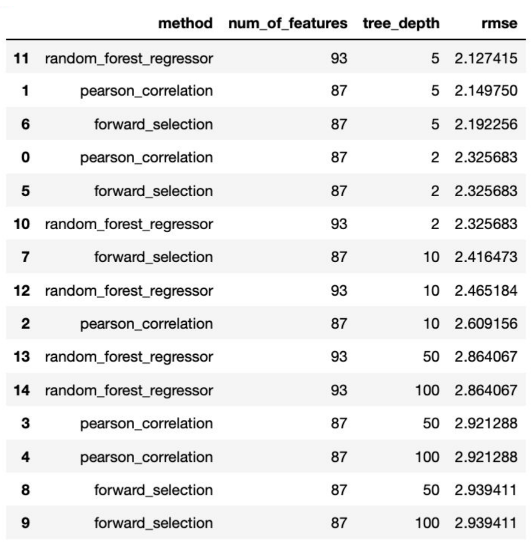
- Decision Tree



	method	num_of_features	tree_depth	rmse
11	random_forest_regressor	91	5	2.316227
1	pearson_correlation	100	5	2.317705
6	forward_selection	100	5	2.360948
2	pearson_correlation	100	10	2.502279
0	pearson_correlation	100	2	2.536599
5	forward_selection	100	2	2.536599
10	random_forest_regressor	91	2	2.536599
12	random_forest_regressor	91	10	2.639008
7	forward_selection	100	10	2.687544
3	pearson_correlation	100	50	2.745791
4	pearson_correlation	100	100	2.745791
8	forward_selection	100	50	3.014796
9	forward_selection	100	100	3.014796
13	random_forest_regressor	91	50	3.031241
14	random_forest_regressor	91	100	3.031241

Fig 15: the result of decision tree

- Random Forest Regressor



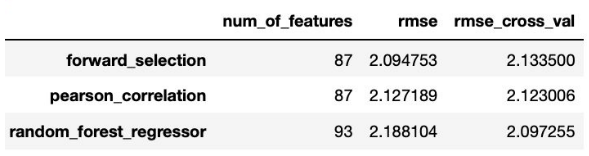
	method	num_of_features	tree_depth	rmse
11	random_forest_regressor	93	5	2.127415
1	pearson_correlation	87	5	2.149750
6	forward_selection	87	5	2.192256
0	pearson_correlation	87	2	2.325683
5	forward_selection	87	2	2.325683
10	random_forest_regressor	93	2	2.325683
7	forward_selection	87	10	2.416473
12	random_forest_regressor	93	10	2.465184
2	pearson_correlation	87	10	2.609156
13	random_forest_regressor	93	50	2.864067
14	random_forest_regressor	93	100	2.864067
3	pearson_correlation	87	50	2.921288
4	pearson_correlation	87	100	2.921288
8	forward_selection	87	50	2.939411
9	forward_selection	87	100	2.939411

Fig 16: the result of random forest regressor

6.2 Evaluation After CrossValidation

We show here the results of each of the models after using cross validation.

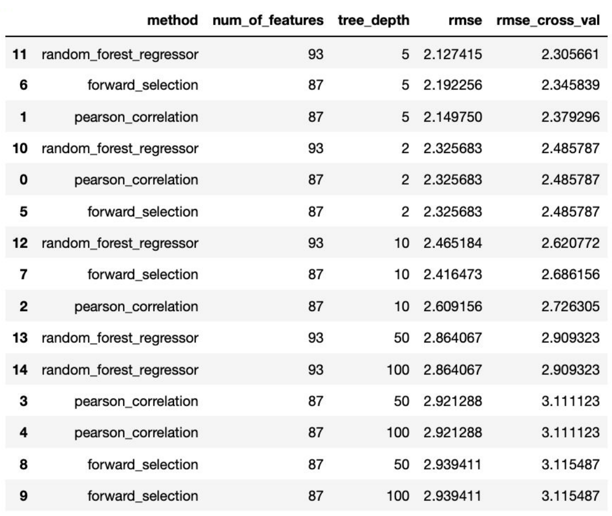
- Linear Regression



	num_of_features	rmse	rmse_cross_val
forward_selection	87	2.094753	2.133500
pearson_correlation	87	2.127189	2.123006
random_forest_regressor	93	2.188104	2.097255

Fig 17: the result of linear regression after cross validation

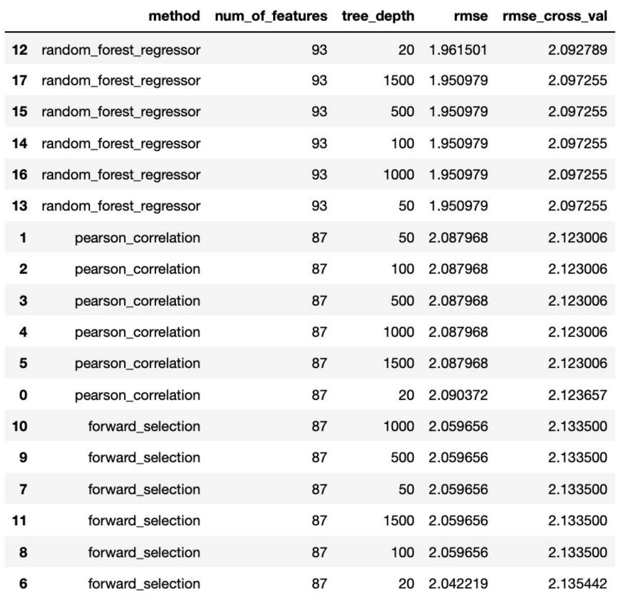
- Decision Tree



	method	num_of_features	tree_depth	rmse	rmse_cross_val
11	random_forest_regressor	93	5	2.127415	2.305661
6	forward_selection	87	5	2.192256	2.345839
1	pearson_correlation	87	5	2.149750	2.379296
10	random_forest_regressor	93	2	2.325683	2.485787
0	pearson_correlation	87	2	2.325683	2.485787
5	forward_selection	87	2	2.325683	2.485787
12	random_forest_regressor	93	10	2.465184	2.620772
7	forward_selection	87	10	2.416473	2.686156
2	pearson_correlation	87	10	2.609156	2.726305
13	random_forest_regressor	93	50	2.864067	2.909323
14	random_forest_regressor	93	100	2.864067	2.909323
3	pearson_correlation	87	50	2.921288	3.111123
4	pearson_correlation	87	100	2.921288	3.111123
8	forward_selection	87	50	2.939411	3.115487
9	forward_selection	87	100	2.939411	3.115487

- Fig 16: the result of decision tree after cross validation

- Random Forest



	method	num_of_features	tree_depth	rmse	rmse_cross_val
12	random_forest_regressor	93	20	1.961501	2.092789
17	random_forest_regressor	93	1500	1.950979	2.097255
15	random_forest_regressor	93	500	1.950979	2.097255
14	random_forest_regressor	93	100	1.950979	2.097255
16	random_forest_regressor	93	1000	1.950979	2.097255
13	random_forest_regressor	93	50	1.950979	2.097255
1	pearson_correlation	87	50	2.087968	2.123006
2	pearson_correlation	87	100	2.087968	2.123006
3	pearson_correlation	87	500	2.087968	2.123006
4	pearson_correlation	87	1000	2.087968	2.123006
5	pearson_correlation	87	1500	2.087968	2.123006
0	pearson_correlation	87	20	2.090372	2.123657
10	forward_selection	87	1000	2.059656	2.133500
9	forward_selection	87	500	2.059656	2.133500
7	forward_selection	87	50	2.059656	2.133500
11	forward_selection	87	1500	2.059656	2.133500
8	forward_selection	87	100	2.059656	2.133500
6	forward_selection	87	20	2.042219	2.135442

Fig 18: the result of random forest after cross validation

7. Conclusion

In general, we observe good results from the done tests. We did some preprocessing analysis in order to clean our data by filling NaN values and normalizing when it was needed. We also have evaluated different encoding solutions such as one hot encoding and binary encoding. Additionally, some feature engineering was applied in order to improve our model.
From the seen results, we observe that the best result was obtained using the Random Forest model. Also, after evaluating different features selection methods, we see the best performance for the random forest regressor method in most of the cases, followed by the pearson correlation and then forward selection.

Finally, in order to see which is the real error of our model, we did a cross-validation evaluation. This shows an increment in the error, but it makes sense because in the first tests we are only using a very small dataset for doing tests. Then our final RMSE is 2.09

References

[1]The 5 Feature Selection Algorithms every Data Scientist should know, Available: https://towardsdatascience.com/the-5-feature-selection-algorithms-every-data-scientist-need-to-know-3a6b566efd2, Accessed on: 26th March,2020

[2] Why, How and When to apply Feature Selection,Available: https://towardsdatascience.com/why-how-and-when-to-apply-feature-selection-e9c69adfabf2 ,Accessed on: 26th March,2020