**Prediction of Movie Success**

**Introduction**

Having some features about a movie we should predict how far will the movie succeed, so we did choose the most important features and the most effective pre-processing techniques and model to get a reasonable accuracy.

**Experimental details**

Starting with pre-processing part, after reading our data we found that we have these types of data (numeric values, strings, json array) we applied label encoding on strings (in our data was the spoken languages ) giving each language a numeric value then coming to the json array we have used the ‘id’ for each element in the json array to be a unique value then we calculated the median for the ‘vote average’ for each id. After doing this step we calculated the summation for the array to get a numeric value.  
Next, we handled the missing data first with median, second with removing the rows for missing data and we got better accuracy with the second experiment. Then we did the normalization step. Finally, we calculated the correlation between the output and each feature to see which features affect the output.

**Results and discussions**

After doing the steps above we get MSE 0.1

**The body**

* Pre-Processing

**Frist transform the data to numerical values**  
  
- Doing label encoding on **‘original\_language**’ feature  
 function “**transform\_map**” is used to return a dictionary of spoken languages  
 mapped into their numerical values

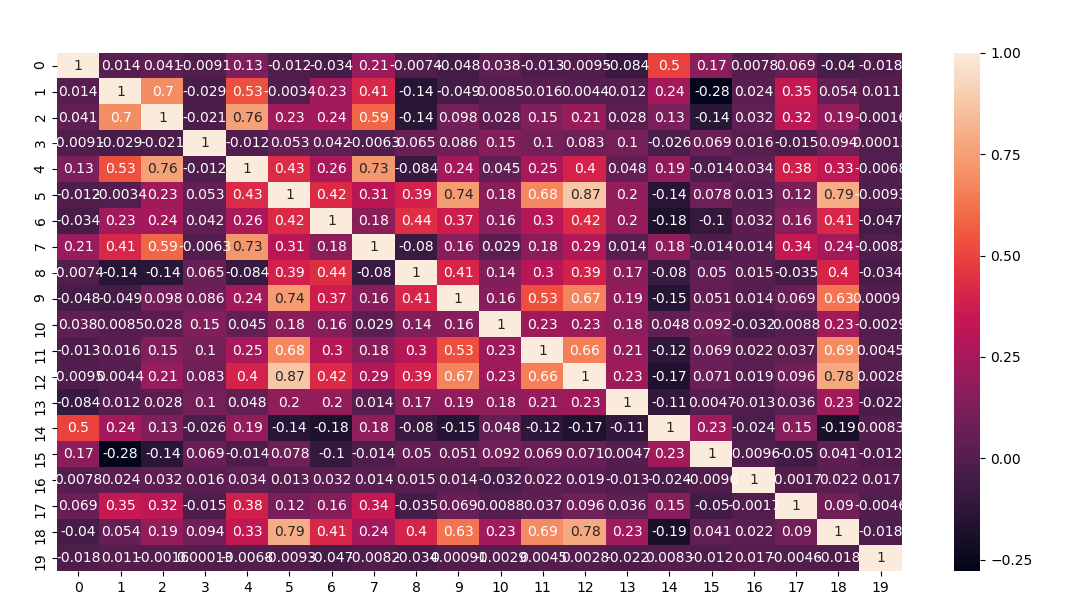
def transform\_map(self, arr):  
 aset = set()  
 dic = dict()  
 for i in range (len(arr)):  
 aset.add(arr[i])  
 counter =1  
 for setElement in aset:  
 dic[setElement]=counter  
 counter+=1  
 return dic

Function “**mapping**” return list of numeric values mapped   
   
 def mapping(self, dic, arr):  
 for i in range(len(arr)):  
 arr[i] = dic[arr[i]]  
 return arr

- Dealing with json arrays  
 function “**getJSONbyID**” return list of arrays of ‘id’ with numeric values or with the parameter Key passed   
  
def getIDFromJSON(self, arr,key,int\_str):

…………………………..  
 return arr  
  
- Putting numerical values instead of ids   
 we calculate the median of vote average for each id / key to be a numerical value instead of the id/key Function “**handlingCatigorialVariables**” return dictionary of ids/key mapped to it’s numeric values   
  
def handlingCatigorialVariables(self,arr,vote,votecount,option):  
 …………………………………….  
 return d  
  
 - Functions **extractmonthFromDate** and **extractYearFromDate**  used to handle the data

**Second deal with missing data** We have tried to ways to deal with missing data   
 1- to Delete missing data   
 2- to fill missing data with mean values for each column   
  
Deleting the data has given better results than filling with mean value  
  
**Third Normalization** we did mean normalization for the data   
 def meanNormalization(self):  
 for i in range(len(self.data)):  
 if i != 5:  
 mx = max(self.data[i])  
 mn = min(self.data[i])  
 meu = np.mean(self.data[i])  
 self.data[i] = (self.data[i] - meu) / (mx - mn)

**Fourth Choosing features** we did correlation between data to see how far the data affect the output and here is the   
 correlation table   
  


* **Models**

1. **Linear Regression**

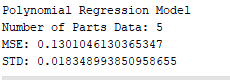
When we use Linear Regression model using cross validation when we split our data to 5 parts we got this results.

**A screenshot of a cell phone

Description automatically generated**

1. **Polynomial Regression**

When we use Polynomial Regression model with degree = 2 and using cross validation when we split our data to 5 parts we got this results .



and when we try to increase the degree the MSE increase .

EX: When degree = 3 results

A screenshot of a cell phone

Description automatically generated

1. **Ridge Regression**

When we use Ridge Regression model with alpha = 0.01 using cross validation when we split our data to 5 parts we got this results .

**A picture containing screenshot

Description automatically generated**

1. **Lasso Regression**

When we use Lasso Regression model with alpha = 0.01 using cross validation when we split our data to 5 parts we got this results .

**A screenshot of a cell phone

Description automatically generated**

1. **KNN Regression**

When we use Ridge Regression model with K = 10 using cross validation when we split our data to 5 parts we got this results .

**A screenshot of a cell phone

Description automatically generated**

**Conclusions**

* Normalizing the output will give the MSE normalized
* Median works on data better than Mean (for json data)
* Deleting missing data has given better results
* Taking features with correlation above 0.3 has given less MS error
* We used cross validation on our data
* models are we use Knn Regression, Lasso Regression, Linear Regression, Polynomial Regression, Ridge Regression and support vector machine regression . the best model gives us better accuracy is KNN model the MSE is 0.1

**References**

<https://machinelearningmastery.com/how-to-prepare-categorical-data-for-deep-learning-in-python/>