Machine Learning Applied to 'Move Hub' City Rating

Authors: Sharon Zhang, Tanita Ressler, Brett Pennoyer, Yana Romalis, Joseph Tinglof

[1] ***Abstract*— This paper seeks to answer the question, "What is the best Machine Learning Model to predict the 'Move Hub' City rating." Linear Regression (Gradient Descent), sklearn Regression, Statsmodel, and KNN (K Nearest Neighbours) are represented as possible solutions in this report. The determined best model is the sklearn regression model. Although all models were close contenders.

I. Introduction

THIS document is an explanation of the dataset 'Move Hub City Ranking'[1] (MHCR), procedure, validation, and comparison of various models applied to MHCR. The goal of this paper is to determine the best machine learning model that accurately and consistently predicts the city ranking of any city in the MHCR dataset. The models implemented are Linear Regression (Gradient Descent), sklearn Regression, Statsmodel, and KNN (K Nearest Neighbours). Each model will be highlighted with a discussion of implementation, validation of model, and output (findings, graphs, output layers). This paper will also address pitfalls in dealing with the MHCR dataset.

II. Introducing the Dataset

The Move Hub City Ranking (MHCR) is an open dataset developed to rate the overall quality of any city searched in their index. The dataset can be found here: https://www.movehub.com/city-rankings/. Their criteria for the overall rating is based off twelve criteria: Purchase Power, Health Care, Pollution, Quality of Life, Crime Rating, Cappuccino, Cinema, Wine, Gasoline, Average Rent, and Average Disposable Income. The output of this model is non-discrete and follows an expected linear graph. For these reasons this group found that any model servicing clustering or involving discrete outputs would not be suitable models for the MHCR dataset. Data models such as linear regression is more applicable. It is this reason narrowing models down to machine learning algorithms with non-discrete vectors is important.

III. Preprocessing

This section will explain the transitioning process of data from csv datafile into various training and testing sets. Before the data could be applied to any model the dataset needs to go through this preparation phase. This step is curtal as it sets the tone and capability of all the models applied. Poorly prepared data could lead to misleading output vectors and inaccurate representations of models. To circumvent this issue, pandas preprocessing libraries were utilized for the preprocessing phase. Pandas offers a robust data frame along with a strong support structure. After importing the data all extraneous data was removed. Such data included city names and column titles. This projects purpose is to predict a numerical rating and city names would add an unnecessary biased and was therefore removed from the input set. Next data normalization was applied to all data. Sklearn normalization 'max' was the method used. This method normalizes to the maximum value of each row; this ensures standard scaling to prevent unwanted biases. The data is then shuffled to create more diverse and consistent model training. The data is also further transformed by splitting ground truth labels and the separation of training, validating, and testing sets. The MHCR dataset has 216 samples, this project divides those samples into 100 training samples, 50 validating samples, and 60 testing samples. The structure output of the preprocessing block is the standard input applied to all models in this project.

IV. Model Procedure

All models meet the requirement of dealing with non-discrete data structures. This parameter was set because of the nature of the MHCR dataset. General procedure, independent of specific models, were to establish plausible application of model, create model, apply data to model, and visualize the output of model. The nature of the MHCR dataset dictates that only model that have discrete outputs will work.

A. Linear Regression (Gradient Descent)

This model, as represented in class, is a linear predicting model. This specific model was chosen because of its ability to be easily applied to non-discrete datasets. Specifically, we thought this model would serve as a good foundation for comparison as it is the first model that we were introduced to in this course. Procedure of implementation was performed in a *class* structure. This was to ensure total encapsulation of code to keep the work space uncontaminated. More specifically to ensure variables were not written over later in other processes, as well as general tidiness. Specific implementation involved first initializing a learning rate (ALPHA), weight vector (THETA), and iteration amount (ITTER). These core parameters are a cornerstone for success with this model. Not choosing the correct combination could lead to a non-converging system. For this reason, we ran this model multiple times with different parameter combinations to produce the best possible outcome. Training involves making a prediction with an input vector, comparing the prediction with the ground truth, and adjusting the THETA weights. These weights are the core learning aspect of this model. Adjusting the weights was accomplished through a gradient descent method. After training the model is run through the testing model and validation to determine the success and

[1]Hove Hub City Ranking Dataset (https://www.movehub.com/city-rankings/)

accuracy of the model. The results of this model are covered in a later section.

B. sklearn Regression

Sklearn Regression is implemented similarly to the Gradient Descent method, but with Sklearn libraries. We implemented this model to make a direct comparison with the self-made Gradient Descent. The following snips of code are the general implementation of our sklearn model.

```
from sklearn.linear_model import LinearRegression
from sklearn import metrics
...
regression_model = LinearRegression(fit_intercept=True)
regression_model.fit(X_train, Y_train)
...
column_names = getColumnNames(raw_data)
...
for idx, col_name in enumerate(column_names):
    print("The coefficient for {} is {}".format(col_name,
regression_model.coef_[0][idx]))
intercept = regression_model.intercept_[0]
print("The intercept for our model is {}".format(intercept))
…
regression_model.score(X_test, Y_test)
```

C. Statsmodel

This model is a package from Python. It allows to explore data, estimate statistical models and perform statistical tests. An list of result statistics are available for each estimator. At first we have to specify an OLS model and fit it. After that we calculated the prediction. In the end we can inspect the statistics derived from the fit.

```
import statsmodels.api as sm
import statsmodels.formula.api as smf

##statsmodel linear regression
model = sm.OLS(Y_train, X_train).fit()

predictions = model.predict(X_test) # make the predictions by
the model

model.summary()
print(model.summary())
```

D. KNN

K nearest neighbours model (KNN) is a not parametric model that can work both with regression and classification problems. In the regression version of KNN, to predict the response for each set of X, KNN averages the values of X's *k* nearest neighbors. Usually the larger the K - the model performs better on a training data (the bias is smaller as K is larger) but increasing K doesn't necessarily improve the model performance on a testing data. SKlearn provides KNeighborsRegressor function that Fits KNN model for regression problems. One of the main parameters that we can send to this function is the K parameters - i.e the number of neighbouring samples that we want to take into account when calculating a response value for each set of X values. Below is an example of general implementation.

```
##Linear Regression, KNN
print("KNN model")
knn = neighbors.KNeighborsRegressor(10)
y_predict_knn = knn.fit(X_train, Y_train).predict(X_test)
regression_model_mse_knn = mean_squared_error(y_predict_knn,
Y_test)

print(math.sqrt(regression_model_mse_knn))
print(" ")
sns.regplot(Y_test,y_predict_knn,data=prepared_data,label='KNN',co
lor='purple')
##END Linear Regression, KNN
```

V. VALIDATION

Validation was a very important step of analyzing the output data in this project.

A. Linear Regression (Gradient Descent)

Validating this model was done with the testing dataset (see PREPROCESSING section to learn more about the various structures the dataset was split into). Validation involved graphing the iteration vs. cost. This shows the convergence of model in each new iteration. Lastly the output layer of the validation was printed and compared in excel to see if the accuracy matched that of the recorded accuracy during testing.

B. sklearn Regression

This model is validated by comparing an output layer with the ground truth label. After validation the testing dataset is applied and plotted. The output can be seen in the MODEL OUTPUTS AND GRAPHS section.

C. Statsmodel

This model is validated by comparing an output layer with the ground truth label. After validation the testing dataset is applied and plotted. The output can be seen in the MODEL OUTPUTS AND GRAPHS section.

D. KNN

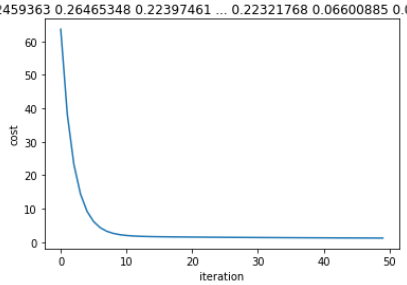
We fitted the model with different K values on the same training set and then tested the results on the same testing set. In the graph depicted in the 'MODEL OUTPUTS AND GRAPHS' section we can further visualize the error associated with different K values.

VI. MODEL OUTPUTS AND GRAPHS

A. Linear Regression (Gradient Descent)

The following is a graph of the iteration vs. cost

```
alpha = 0.001  theta = [[0.14397086 0.3058133  0.25880791 ... 0.25793326 0.07627477 0.05671094]
 [0.11082994 0.23541758 0.19923245 ... 0.19855913 0.05871694 0.04365655]
 [0.12040599 0.25575837 0.21644673 ... 0.21571525 0.06379026 0.0474286 ]
 ...
 [0.12642478 0.2685431  0.22726637 ... 0.22649832 0.06697898 0.04979944]
 [0.12672497 0.26918074 0.227806   ... 0.22703613 0.06713802 0.04991769]
 [0.12459363 0.26465348 0.22397461 ... 0.22321768 0.06600885 0.04907814]]
```



The accuracy, after running this model multiple times with different starting conditions, is 94%. Suspicion of the 6% error was thought to be sample size. Convergence might have been higher if there were more than 216 samples to train on.

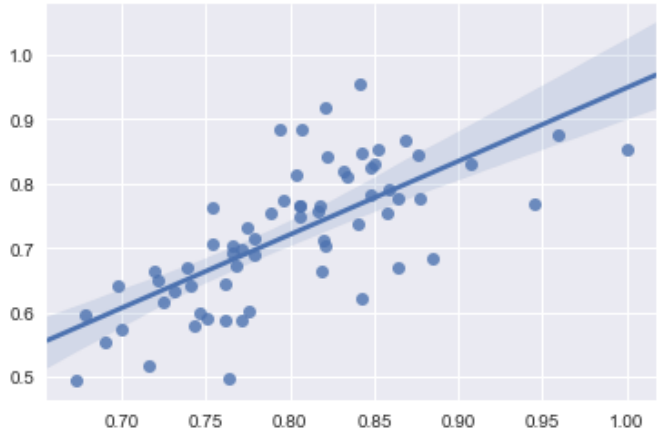


Figure 1. *depicts the comparison of the ground truth [X-axis] and the predicted [Y-axis], The shaded region represents confidence interval region. From this graph we can determine how well this model did on a individual level.*

Mean Absolute Error: 0.06442134738561392
Mean Squared Error: 0.006807743786113957
SQRT(Mean Squared Error:) 0.08250905275249472

B. sklearn Regression

The coefficient:

Purchase Power:	0.21944076410106284
Health Care:	0.038118307521712365
Pollution:	0.01769791852489193
Quality of Life:	0.022151725137261803
Crime Rating:	0.026941688780527027
Cappuccino:	-0.06578192394047244
Cinema:	0.6750833956826596
Wine:	-0.011215817939543916
Gasoline:	0.02498096558986842
Avg Rent:	0.1514625239922436
Avg Disposable Income:	-0.10831061044712852
The intercept:	0.5900368332130035

Mean Absolute Error: 0.03274272830808498
Mean Squared Error: 0.006539819668310729
Root Mean Squared Error: 0.08086915152461245

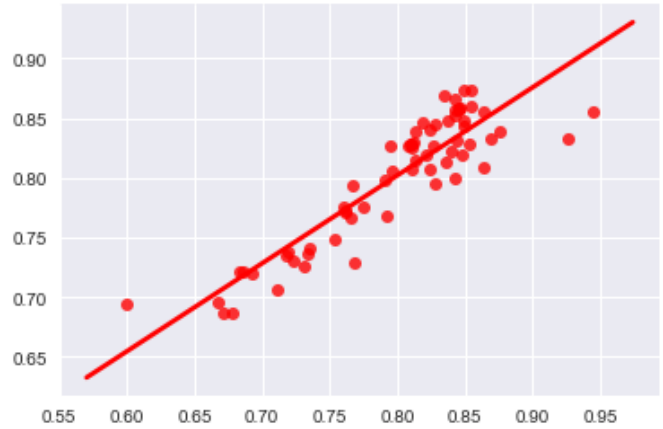


Figure 2. *depicts the comparison of the ground truth [X-axis] and the predicted [Y-axis], The shaded region represents confidence interval region. From this graph we can determine how well this model did on a individual level.*

C. Statsmodel

The following is a list of the result statistics, which include the coefficient ("coef").

OLS Regression Results

Dep. Variable:	y	R-squared:	0.992
Model:	OLS	Adj. R-squared:	0.991
Method:	Least Squares	F-statistic:	1020.
Date:	Mon, 30 Apr 2018	Prob (F-statistic):	1.47e-88
Time:	17:47:10	Log-Likelihood:	121.89
No. Observations:	100	AIC:	-221.8
Df Residuals:	89	BIC:	-193.1
Df Model:	11		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
x1	0.3035	0.088	3.451	0.001	0.129	0.478
x2	0.0201	0.063	0.318	0.752	-0.106	0.146
x3	0.1355	0.029	4.724	0.000	0.079	0.193
x4	0.3863	0.073	5.270	0.000	0.241	0.532
x5	0.1934	0.038	5.092	0.000	0.118	0.269
x6	0.0966	0.083	1.163	0.248	-0.068	0.262
x7	0.3311	0.563	0.588	0.558	-0.787	1.449
x8	0.2870	0.073	3.915	0.000	0.141	0.433
x9	0.2742	0.047	5.807	0.000	0.180	0.368
x10	0.4765	0.099	4.806	0.000	0.279	0.674
x11	-0.5611	0.106	-5.306	0.000	-0.771	-0.351

Omnibus:	0.817	Durbin-Watson:	2.126
Prob(Omnibus):	0.665	Jarque-Bera (JB):	0.913
Skew:	0.199	Prob(JB):	0.634
Kurtosis:	2.753	Cond. No.	119.

The following lists the results of Mean Absolute Error (absolute differences between prediction and actual), the Mean Squared Error (differences between prediction and actual)and Root Mean Squared Error (average of squared differences between prediction and actual)

Mean Absolute Error: 0.0540157161767
Mean Squared Error: 0.00474295609438
Root Mean Squared Error: 0.068869122939

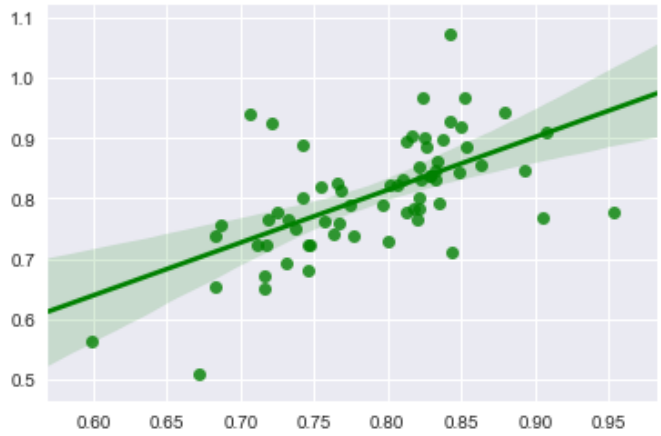


Figure 3. *depicts the comparison of the ground truth [X-axis] and the predicted [Y-axis], The shaded region represents confidence interval region. From this graph we can determine how well this model did on a individual level.*

D. KNN

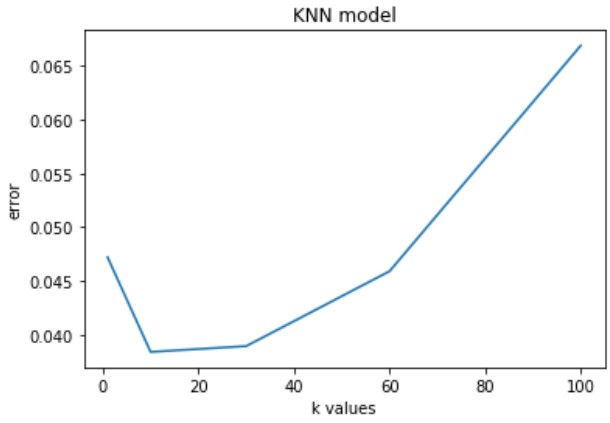


Figure 4. shows that when KNN value gets large, the performance of the model actually got worse. The KNN value would be under 20.

Mean Absolute Error: 0.028878181818181806
Model Mean Squared Error: 0.0016939576939393936
Model SQRT(Mean Squared Error): 0.04115771730720004

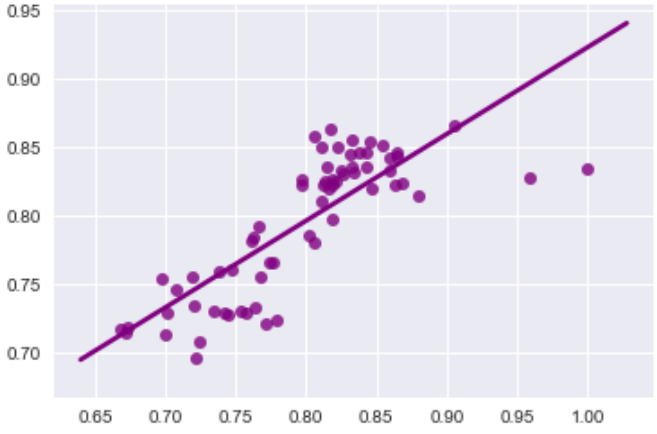


Figure 5. *depicts the comparison of the ground truth [X-axis] and the predicted [Y-axis], The shaded region represents confidence interval region. From this graph we can determine how well this model did on a individual level.*

VII. COMPARING MODELS

A major goal of this project is to compare the models introduced in the INTRODUCTION. This section will seek to not only rectify, but clearly determine the best model to use for Move Hub City Ranking (MHCR).

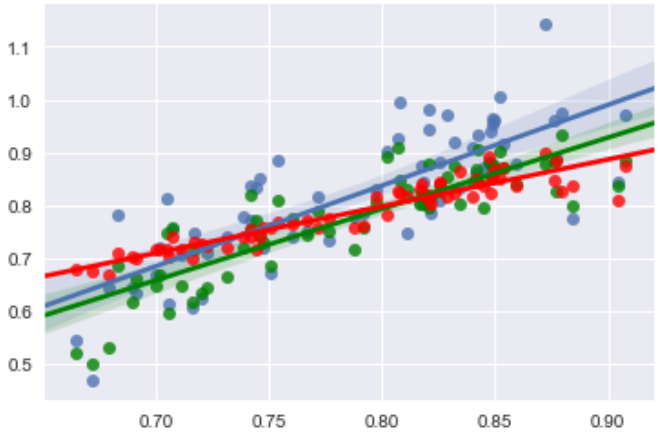


Figure 6. Overall comparison of all the above models, and the straight lines represent the best fit lines.

	Gradient Descent	SKLearn	Stat Model	KNN
Mean Abs. Error	0.0835	0.0242	0.0552	0.0289
Mean Squared Error	0.0108	0.0010	0.0049	0.0017
SQRT(Mean Squared Error):	0.1037	0.0321	0.0699	0.0412

Table 1: Overall error comparison of all above models.

After we compared all the results of the mean absolute, mean squared and root mean squared error we found out that the best model with the smallest error is SKLearn regression. The smallest error means that this model has the best accuracy in prediction.

VIII. GENERAL ISSUES AND SOLUTIONS

This project had many obstacles and this section will clarify common pitfalls associated with the Move Hub City Ranking (MHCR) dataset. Firstly, due to the nature of the dataset, clustering models should be avoided as they serve no insight into solving this dataset. Iteration and refinement is key. The first model testing is not always the best combination of parameters to elicit a proper representation of data in the model. Meaning, starting conditions of models should be

adjusted to maximize output convergence. Best practice dictates that the first attempt will most likely not be the best output. Iteration of the same model with different starting conditions should be exercised when dealing the MHCR and all other datasets. Data visualization and comparing models was also a challenge. It is hard to find the correct way to display the models and compare results. We ended up solving this issue by focusing on accuracy and error graphs.

IX. Conclusion

In summation, this project implemented Linear Regression (Gradient Descent), sklearn Regression, Statsmodel, and KNN models to find the best method of predicting the Move Hub City rating (MHCR dataset). We compared the models and determined that the), sklearn Regression is the best method in predicting the city model. The models tested were all very close in accuracy however it is clear that the) sklearn Regression model is the best option. This is due to the convergence and accuracy factors compared to the rest of the models. On average, this model performed consistently better and therefore is our pick of best model for this dataset.

X. Contributions

In general, each member is responsible for reporting on their system, model, and methods in this report.

Brett Pennoyer – Data Preprocessing, main.py, linear regression model (Gradient Descent), graphing, data visualization. Report Writing and construct according to IEEE format.

Tanita Ressler – Data Preprocessing including Normalization, Linear Regression (Gradient Descent, Statsmodel, and KNN). Model visualization and performance metrics, Report General Writing.

Yana Romalis - KNN model, Graphing, Report General Writing

Sharon Zhang - Preliminary Data Processing and acquisition, SKLearn Linear Regression including data visualization and performance metrics. Model comparison and General Report Writing.

Joseph Tinglof - Code Integration, Data Visualization, Report Formatting and Coordination, General Report Writing.