RESULTS OF FEATURE IMPORTANCE

SAMI KHAN

02/14/2021

Contents

1.0 SUMMARY

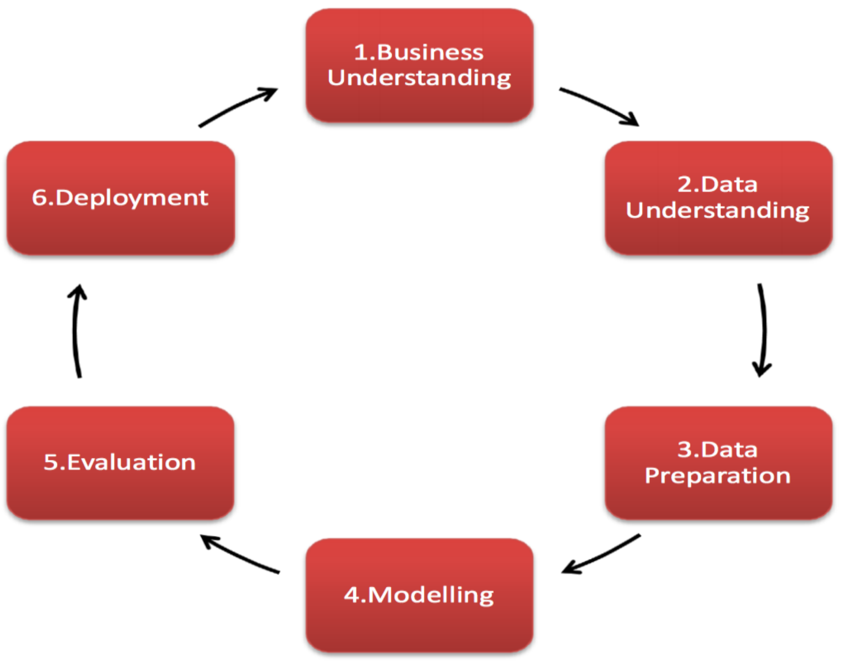
This document summarizes my methods and results for the feature importance assignment. I was given a sample data set highlighting the possible factors that influence the energy burden in a Census Tract . The data set consisted of 1957 unique data points (rows) of different Census Tract for years 2013-2016. It had 16 different exogenous (columns/features). However, we can drop id2 since geography encaptures the same information. The energy burden (target) was calculated by dividing the total energy bill by the median income . No test set was provided so 20% of the data set was split into test set. The objectives were to develop a model that could predict the importance of a feature that the energy burden depends on.

This task is a regression task since we are trying to calculate the relationship between the target variable and the input variable. Hence, if there is a relationship between the input factors and energy burden, any classical regression algorithm could be applied to this problem. I applied several classical Machine Learning Regressors (e.g Decision Tree Regressor, Random Forest Regressor, Extreme Gradient Boosting) and a Dense Neural Network to map the relationship between the input features and energy burden. The Random Forest Regressor performed almost the same as the Decision Tree Regressor and since Decision Trees are prone to overfitting, I included the Random Forest Regressor instead. All three algorithms that I picked performed good, however, if I had to pick one, **I would choose the Deep Neural Network since it performed well on both, combined states and individual states**. The Extreme Gradient Boosting and Random Forest Regressor overfit when data points for the state of Colorado were chosen.

The following sections further explain the techniques employed for pre-processing, hyper-parameter tuning, model training and selection.

2.0 PRESENTATION

The standard medium for data science development and presentation is jupyter notebook for many companies. This solution was also started and executed for the with a jupyter notebook but as the code grew, I decided to also package it as an app. This was to allow for more flexibility in terms of modularizing the code, readability, and architecture simplicity. It also provides the ease of collaboration while using standardizations like PEP8. I used Pycharm IDE that allowed me to debug efficiently and automatically convert my script to follow PEP8 requirements. Packaging it as an app also gives the flexibility to apply multiprocessing as the training set size increases and can also be used to run on a Multi-GPU PC with the help of Tensorflow as well if ever needed. I have attached my code with the requirements.txt for dependencies, and a readme.md for instructions to run the code. I have also provided the results of every stage of my analysis in this report. The diagram below highlights the general workflow used for all data science projects and I tried to follow it to the best of my abilities.



3.0 DATA PREPROCESSING

The first steps in every machine learning modeling pipeline always include data exploration and pre-processing. Upon exploration of this dataset, I discovered that it contained missing data, categorical data, erroneous data (values were negative), and data of varying units. This category explains the techniques and results, where applicable, used to preprocess the data.

3.1 MISSING DATA

Of the 16 feature columns in the training data, 6 columns contained missing data (null and unknown);

Column	**No. of missing values**
percent_asian	1
percent_black	1
percent_hispanic	1
percent_multifamily_housing	6
median_income	12
asthma_rate	1634

There are multiple ways to handle missing data. A simple technique will be to simply drop all affected rows. After removing affected rows, I still had 1824 remaining rows which is about 93.25% of the original data set. The general rule to drop affected rows is to drop the rows if rows with missing values are in the ballpark of 5%-7% and since our data lies in that range, I decided to get rid of it. Another technique will be to regress/predict the missing data using the other available features. That will however be tedious and maybe overkill for this exercise. Hence, I opted to remove the rows with missing values altogether.

3.2 DROPPING COLUMNS

I decided to drop some irrelevant columns since they were either redundant or had no correlation with the target variable. Asthma_rate was also removed since 85% of its values were missing so it did not make any sense to impute the missing values. Redundant columns included id2 since it was essentially the same thing as geography, year and state because they had no correlation with the energy burden. It is important to point out that these columns were dropped manually while in the future, it is better to remove columns based on the results from a feature selection algorithm such as Recursive Feature Elimination that would result in relevant columns only.

3.3 CATEGORICAL DATA

There were non-numerical data present in the dataset. While majority of them were removed due to irrelevance, geography was feature engineered into County and State. From there, it was one-hot encoded which resulted in three new indicators corresponding to the three County, State combinations in the data set. One hot encoding was an important step in order to be run by the machine learning algorithms.

3.4 SCALING

The scales of the various information type varied widely. For example, the average cost columns had several negative values. I took care of them by converting them into nan values. The columns were converted to the same units (decimals) since some columns existed as percentages. Feature scaling was a key step in my machine learning pipeline because most machine learning algorithms require similar scaling to learn the right cost function. Decision trees, random forest and extreme gradient boosting are some of the few machine learning algorithms that are scale invariant since these models only need to pick “cut points” on features to split a node. Splits are not sensitive to monotonic transformations: defining a split on one scale has a corresponding split on the transformed scale.

However, for my deep neural network, I had to perform scaling. Normalization and standardization are the two most widely used scaling techniques. I chose standardization for my purpose because for gradient descent type optimizers, having a mean of 0 and a standard deviation of 1 makes it easier to learn the weight, hence speeding up training. Also, standardization maintains useful information about outliers while making the algorithm less sensitive to them in contrast to min-max techniques like normalization.

3.5 FEATURE EXTRACTION/SELECTION

Selecting only the features that are most relevant to predict the outcome or mapping the features to a higher dimensional space can be very useful to speed up training, and to model nonlinear (more complex) relationships. This is especially true for conventional machine learning techniques like linear regression. This is not the case for deep neural networks because the learned weights can turn on/off the effect of a feature based on its prediction relevance therefore acting as an automated feature selector/extractor within the model. Furthermore, regularization techniques can also be used to further control the extent of this. Nonetheless, the feature extraction/selection method can be useful to speed up training for a neural network. Thus, I implemented a feature extraction technique called Recursive Feature Elimination to extract relevant feature to be fed into the machine learning model for both conventional algorithms and deep neural network.

3.6 TRAINING TEST SPLIT

I used a 80%-20% training and testing split. The model can be further experimented with various distributions of train/test split.

4.0 CONVENTIONAL MACHINE LEARNING

I decided to build pipelines of various two classical machine learning models.

4.1 TRAIN AND TEST ACCURACIES

The table below shows the baseline results of the implemented methods with no advanced parameters using 20% test set. It was either trained on data points for both states or data points for each state. It is to be noted that different input variables, mostly selected through recursive feature elimination, were considered for each state.

ALGORITHMS	**STATE**	**TRAINING ACC. %**	**TESTING ACC. %**
Random Forest	Colorado and Georgia	99.1	91.3
Extreme Gradient Boosting	Colorado and Georgia	1.0	93.2
Random Forest	Colorado	98.0	57.5
Extreme Gradient Boosting	Colorado	1.0	51.5
Random ForestRandom Forest	Georgia	99.2	85.6
Extreme Gradient Boosting	Georgia	1.0	89.7

5.0 DEEP LEARNING

For a relatively simple problem and dataset like the one provided, one will expect a much better performance than those of the conventional machine learning regressor. Hence, I built a simple 5-layer neural network to fit the data. The metric I used to track neural network performance was mean absolute error. An ideal MAE is not 0 since then you would have a model that perfectly predicts your training data, meaning that it is very unlikely to predict any other data. We want the to be as close to 0 as possible i.e. very low. The very first model I built resulted in a MAE of 0.0021 on test data showing low generalization error. Thus, I chose not to further optimize the model considering this is not for production purposes.

5.1 DNN MODEL DESCRIPTION

See below for a description of the model used to predict the test set;

- Feature Extraction: None
- Regularization: None
- Number of layers: 5
- Nodes: [500, 500, 500, 250, 25]
- Optimizer: Adam
- Activation (hidden): rectified linear unit
- Batch Size: 64
- Loss: mean squared error
- Epochs: 500
- Train/Test split: 80/20

6.0 CONCLUSION

The neural network was selected as the best model due to its consistently low mean squared error across different combination of states. I believe the model can still be improved considering that I am yet to tune the hyperparameters. Typically, metrics like mean squared error, and mean absolute error provide insight for selecting model improvement techniques for a regression problem while parameter search (e.g. grid search or random search) can be used to find better hyperparameters.

The median income, cost of electricity and cost of gas are the most important indicators for 'energy burden' respectively as displayed by all algorithms. This is confirmed by Recursive Feature Elimination that was performed earlier. I prefer the average feature importance of all models for this particular combined data set (both states) because all three algorithms indicate good performance reflected by their metrics.

When Census Tract only belongs to Colorado, Random Forest and XGBoost display low performance on test set and high performance on the training set which means that both of the models are overfitting. Neural Network results in a low mean absolute error, so I would pick the neural network approach for the case when state is "Colorado". Therefore, avg_cost_electricity has the highest feature importance.

When Census Tract only belongs to Georgia , all the machine learning algorithms perform very well like the combined state model. It has similar results with the combined states largely due to the fact that majority of the Census Tracts come from Georgia. Median income is the most important feature.