USA Baby Name Analysis

Chaozhang Huang
ch616@rutgers.edu

***Abstract*—Names are important. People are given names since they were born for various reasons: to honor a memorable person, to express auspicious expectation to their descendents or to emphasize the uniqueness of their personalities. Names are also representation of people in the social network. Others can obtain a first impression from one's name before actually knowing them.**

I. Introduction

The purpose of this project is to analyze the naming of US citizens from 1910 to 2014 over 50 different US states and District of Columbia. The goal of this research project is to develop a prediction model for name ranking based on machine learning techniques using the correlation between the choice and frequency of name on different time period. Analysis on the same dataset based on geological locations, special event, etc will be performed if time allowed.

Once the original data was obtained, it can be processed to fit the need of this research using distributed computing with spark and zeppelin. The resulting data could be visualized to show how the preference of name choice change as time passes.

The next step is to train a model that gives prediction of name counts of a specific year based on data of the past. It will be using name counts data in the past as features to predict future naming, so that we could obtain a ranking of each year's name choice based on past evidence. We would like to train a model that predicts top-10 most popular name accurately using name counts in the past.

Through analyzing the USA name data, we could learn how is the popularity of name change over time. We can obtain an model that can predict future ranks or popularity of a specific name. This model will be useful for future investigation on how time, event, geological location affect the name choice of Americans. Using this model, we would be able to build a practical name recommender that takes realistic evidence into account. The focus of this project is to build a reliable model. The recommender system would be developed in the future if time does not allowed.

II. Problem Definition

As what people need to bear with their whole life, deciding a good name is very important. However, naming can sometimes be complicated and needs to take many things into consideration: what personalities are reflected by the name, does it reflect a sense of family bond, will it be too popular, etc. To solve this problem, this project aims to train an effective model to predict popularity of names in the future.

Id	Name	Year	Gender	State	Count
1	Mary	1910	F	AK	14
2	Annie	1910	F	AK	12
3	Anna	1910	F	AK	10
4	Margaret	1910	F	AK	8

Fig. 1: Snippet of StateNames.csv

Id	Name	Year	Gender	Count
1	Mary	1880	F	7065
2	Anna	1880	F	2604
3	Emma	1880	F	2003
4	Elizabeth	1880	F	1939
5	Minnie	1880	F	1746

Fig. 2: Snippet of NationalNames.csv

A. Time series vs Supervised learning

The dataset after preprocessing could be seen as a time series that depicted name popularity over years with shape of (105, 93805). However, this dataset can not be directly used to train the model since each new state of time series problem requires information from previous state, we need to store data from previous state for each target. i.e. for each target vector y_i at time t, we use y_{i-1} at time $t-1$ ... y_{i-k} at time $t-k$ to train the model. Thus, we need to convert the time series data to input of supervised learning problem. Suppose we want to train the model with k previous year's data as features, we need to transform the input to a 3D tensor with shape (105-k, k+1, 93805) such that we will have (105-k) instances of the supervised learning problem, each has k vectors of length 93805 as training data, and 1 vector of length 93805 as target data. An algorithm to convert time series to a supervised learning problem can be found online [1].

III. Data Acquisition and Preprocessing

A. Data Acquisition

The data[2] of this project is found on "kaggle.com". It is a collection of government-published name data of applications for a social security card from 1910 to 2014. The data set contains two files in CSV (Comma-separated values) format:

- StateNames.csv: This file contains name counts for each of the 50 states and District of Columbia from 1910 to 2014. In order to protect privacy, name that has less than

5 occurrences then it will be neglected. Each record has the following format:
  - 2-digit state code,
  - sex (M = male or F = female),
  - 4-digit year of birth (starting with 1910),
  - the 2-15 character name,
  - the number of occurrences of the name.

  A snippet of data in StateNames can be found on Figure 1
- NationalNames.csv: This file contains counts for different names nationwide. In order to protect privacy, name that has less than 5 occurrences then it will be neglected. Each record has the following format:
  - 4-digit year of birth (starting with 1879),
  - sex (M = male or F = female),
  - the 2-15 character name,
  - the number of occurrences of the name.

  A snippet of data in NationalNames can be found on Figure 2

Due to time limitation, in this project, we will only focus on analyzing and training the model based on NationalNames.csv, experiment and analysis on StateNames.csv could be done in future.

B. Data Preprocessing

1) Data Cleaning: For the use of this project, the data needs to be preprocessed to suit the needs for model training and classifying.

First of all, the data from two separate files did not match in time(StateName.csv starts in 1910, NationalNames.csv starts in 1879), there is a need to filtered NationalNames.csv to match the time with Data in StateNames.csv for future analysis. For this step, we simply truncated data in NationalNames.csv to start with 1910.

Also, in this project we did not focus on gender-wise information. Thus gender information was not considered during training process. Thus, if a name involves two genders, we will combine them into one entry in the dataset. For example, if there are 50 men and 200 women named 'Emma', we would combine them to one entry ('Emma', 250) in the database.

Sample data after this process can be found in Figure 3

Name	Year	Count
Mary	1910	22848
Helen	1910	10479
Margaret	1910	8226
Dorothy	1910	7318
Ruth	1910	7212

Fig. 3: Cleaned Data

2) Weighted Data Transformation: Textual data often exhibits a power law distribution, name counts have no exception. Small amount of popular name choice covers most of names. To alleviate this effect and make the model more vigorous, we scaled data using log-transformation when training model. The comparison between regular TF(term frequency) and log-TF would be shown in section 3.3.

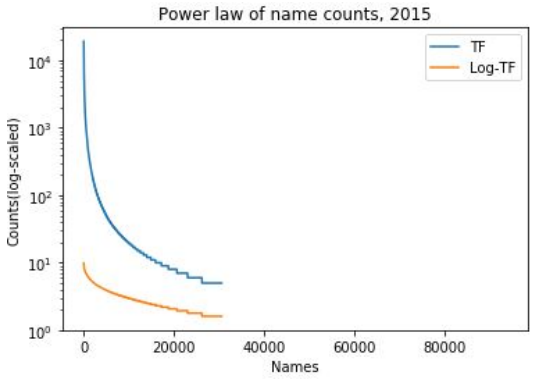

Fig. 5: Power-law-like distribution on name counts, year of 2014

C. Sample Data Visualization

The dimension of NationalNames.csv(NN) dataset is (105, 93805), which means that the dataset covers counts of 98305 unique name over 105 years(1910-2014). Assuming no new words would be made to be name in the future, we can categorize each unique name as a feature to train the model. Thus, from the data set we observed 105 instances with 93805 features.

Name	Count
James	5129096
John	5106590
Robert	4816785
Michael	4330805
Mary	4130441

Fig. 4: Top 5 most popular name of all time

Figure 4 shows a sample visualization of the data: Among all the available names from 1910 to 2014, "James", "John", "Robert", Michael" and "Mary" are the most popular names from all records from 1910 to 2014. This snippet of visualization is obtained by performing MapReduce on the original data set. Part of analysis and processing of Data in this project are done on Zeppelin using Spark 2.

The Power-law-like distribution of name count were shown in Fig 5. We can see that few most popular names covered most part of the dataset. To alleviate this phenomenon, we applied weighted transformation to the data, making names with higher counts less aggressive. In Fig 5, the Y-axis is name counts(log-scaled) and the X-axis represents index of unique names sorted by the corresponding name counts. Note that many 'unique' names were deprecated by the time of 2014 thus there has no data beyond ˜30000 in Figure 5.

A much straightforward comparison between raw data and weighted data was shown in Fig 6 and 7, where Fig 6 shows

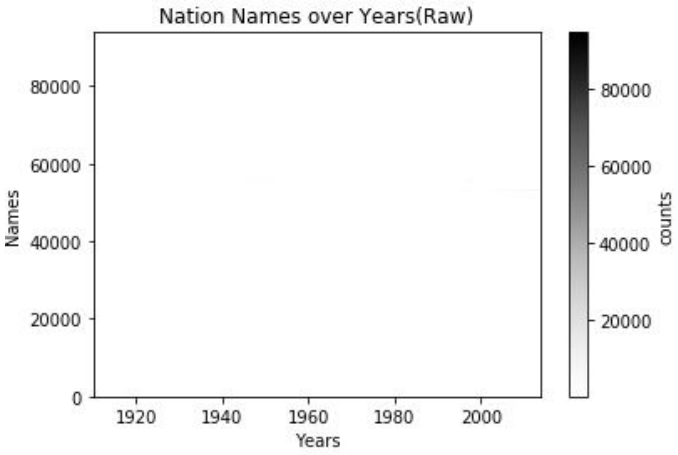

Fig. 6: Raw Name Frequency Density

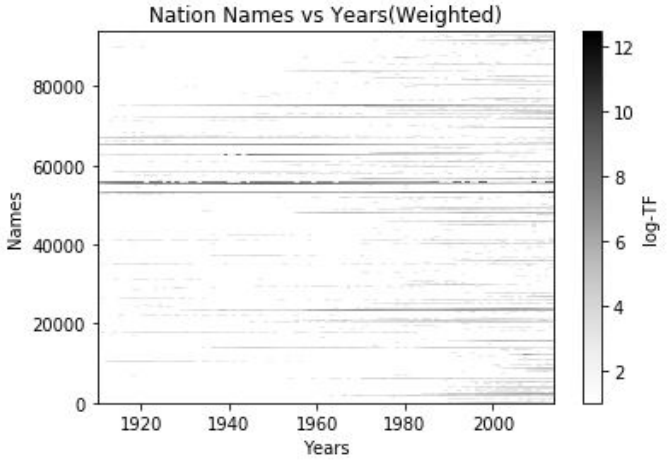

Fig. 7: Weighted Name Frequency Density

the name count density over 105 years and Fig 7 shows the weighted name count density over the same time frame.

IV. Multivariate Model

A. LSTM-RNN Model

RNN(Recurrent Neural Network) is a neural network that takes advantages of back propagation. Having multiple layers of node, each neuron in layer i has a directed connection to every other node in the next layer $i+1$. Upon reaching the last layer, a loss function was computed using the final output from the model and weights of each nodes in every layer was updated using back propagation for gradient descent to minimize the loss function.[3]

Since basic RNN suffers badly from the gradient vanishing problem[4], sophisticated improvement to the RNN model was made to overcome this issue. One of them is called LSTM(Long Short Term Memory) network, an RNN that composed of LSTM units. Each LSTM has an input gate, an output gate, and a forget gate, allowing the unit to control the flow of information of the cell, thus alleviating the gradient vanishing problem.

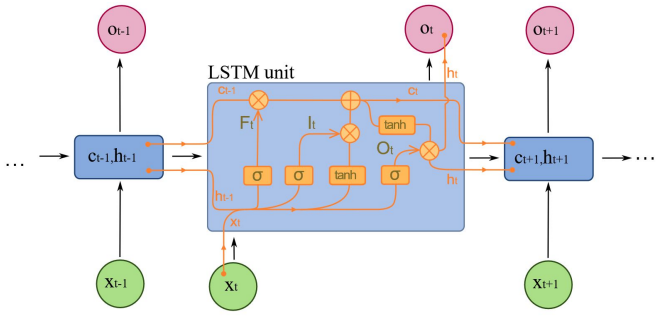

Fig. 8: Example of LSTM based RNN. By François Deloche[8]

Fig 8 showed an example flow graph of LSTM based RNN model. In this project, the LSTM network was implemented as follows:

- X_i is a input tensor of size (k, number of features) at step t_i where k is the number of years we want to consider as 'lag' observations.
- (C_i, H_i) represent state at step t_i
- O_i would be a output vector of length (number of features) at step t_i

RNN(Recurrent Neural Network) with LSTM(Long Short Term Memory) are known to be well suited for processing time series data. In this project, we used RNN model provided by Keras[5], a high level neural network API on Python that ultimately use TensorFlow[6] on GPU to train the model. The implementation of this model was inspired by [7].

B. Feature Filtering

From Figure 5 it is clear that out of 93805 unique names over the 105 years interval, there is a large portion of them were either deprecated or vanished over time, leaving empty tail in the power-law graph. When training the model, we could filtered out some of the irrelevant features, resulting in faster training performance and more relevant accuracy of prediction.

1) Variance Filter: The variance of a given dataset can be computed using the following equation:

$$s_N = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2}$$

It measures the extents where a set of numbers spread out from their mean value. If the variance of a set of data(counts of a name over years, for example) is small, then this set of data did not fluctuate much around the mean. In order words, if the variance of a certain feature is small, then it might be save to neglect it because it did not change much over time. Using the variance filter, we could filtered out part of the irrelevant features under a certain threshold.

2) Mean Filter: The mean of a given dataset can be computed using the following equation:

$$\bar{x}_N = \frac{1}{N}\sum_{i=1}^{N} x_i$$

It measures the overall amplitude of a feature. If the mean of a set of data(a feature, for example) is small, then it's overall amplitude is small. In order words, if we rank the importance of features by it's relative amplitude, then the features with small mean would be trivial thus save to remove. By ranking the features in the order of mean frequency, we could filtered out part of the features that were under a certain threshold.

C. Feature Selection

After applying above filters to the dataset separately, we obtained two list of features to remove:

- Features to remove due to low variance.
- Features to remove due to low frequencies.

For the final step of feature selection, we removed features belong to the intersection of above two sets. The rational behind this decision are as follows: What we want to remove is features that not only have low variance but also low mean frequency. Fig 7 showed that an important feature with relatively low variance might exist(an almost constant dark line below 60000). If we only remove features with low variance, this kind of important features would be deleted as well. Similarly, there are features with relatively low means but larger variance, taking the intersection of two sets solve this problem nicely. Leaving a data set with relevant features to train the RNN model.

V. Independent Univariate Model Set

Through analyzing the dataset, it is clear that this problem become a typical "$p >> n$" problem where p is the dimension of feature space and n is the available observations. Fitting the model with limited amount of samples when trying to predict multiple multi-dimensional target values often suffers from the curse of dimensionality, which means that we might need to feed an overwhelming amount of training data to the model before we could get meaningful output from it. To reduce the effect brought by high dimensionality, we make the following assumption: Frequency of each name is independent of other names. With the above assumption, each name i can be trained with an independent model m_i that using it's own frequency count over years as observation to model m_i to predict the frequency data of this specific name.

VI. Evaluation

A. Experiment Setup

The experience of this project was conducted on a PC platform qquipped with an i7-6700 CPU (4 cores, 3.4Ghz), 16GB of Memory and an Nvidia GTX1080 GPU with 8GB of memory. Part of sample data visualization and analysis were performed on Apache Zeppelin with Spark2. The model was implemented on a Jupyter notebook using Python 3.6 with following package used:

- Keras[5]
- TensorFlow[6]
- Pandas[9]
- Scikit-learn[10]
- Scipy[11]

The dataset was empirically split into 63.3% of training set and 33.7% of test set.

B. Performance Metric

We designed a scoring metric for prediction of top-10 most popular names over the years. The accuracy scoring metric is as follows:

Rank	Score	Rank	Score
1	0.25	6	0.06
2	0.20	7	0.05
3	0.15	8	0.04
4	0.11	9	0.03
5	0.09	10	0.02

TABLE I: Placement Score for Correct Top-10 most popular Name Choice Prediction

Rank	Score	Rank	Score
1	0.02	6	0.02
2	0.02	7	0.02
3	0.02	8	0.02
4	0.02	9	0.02
5	0.02	10	0.02

TABLE II: Present Score for Correct Top-10 most popular Name Choice Prediction

The total score of a prediction is given by:

$$0.8 * \sum Placement_i + \sum Present_i$$

In other words, the accuracy score of a top-10 prediction will be determined by the 0.8 times the sum of placement score and 0.2 times the sum of present score.

Placement score of i_{th} name of a top-10 prediction was assigned if the i_{th} name is correctly predicted and is in the right location.

Present score of i_{th} name was assigned if the name was correctly predicted by the prediction(either misplaced or in the right location).

C. Experiment Result of Multivariate LSTM Model

Dataset	#Features	Training Time	Speedup
Original	93805	2 min 56 s	-
Reduced	35049	1 min 9 s	x2.55

TABLE III: Comparison between Training Time w/ or w/o feature selection

Table III showed the speedup of model training with or without feature selection. Using the feature selection techniques we were able to reduce the number of features from 93805 to 35409, resulting in a similar reduction on training time with up to 2.55x speedup.

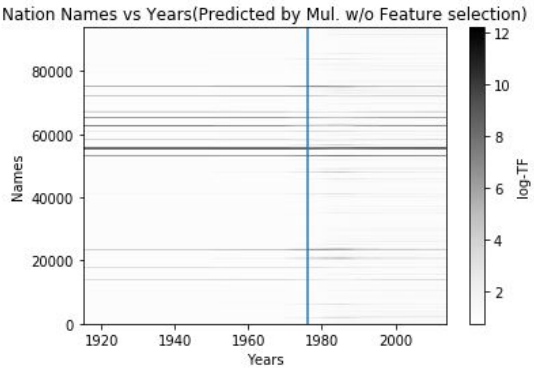

Fig. 9: Predicted Density of Name Frequency by Multivariate Model w/o Feature selection

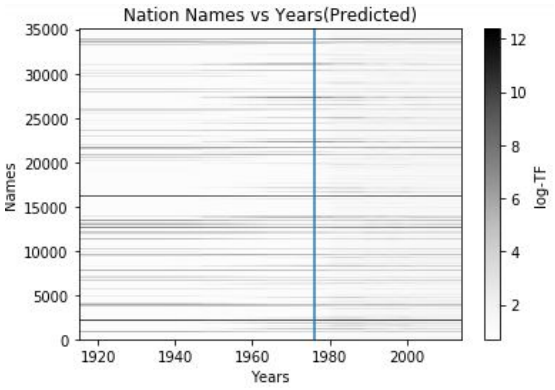

Fig. 10: Predicted Density of Name Frequency by Multivariate Model w/ Feature selection

Figure 9 and 10 showed a comparison between the prediction result on training set and test set on the multivariate model trained by using original dataset and feature-selected reduced dataset. Predictions before the blue line are made from training set, where those after the blue line are made from test set.

Through the visualization, its clear that the reduced dataset not only neglect the irrelevant features(blank in the graph) but also captured more details than the original dataset in terms of training. As a result, the accuracy score of feature-selected model is higher and better for most of the prediction made.

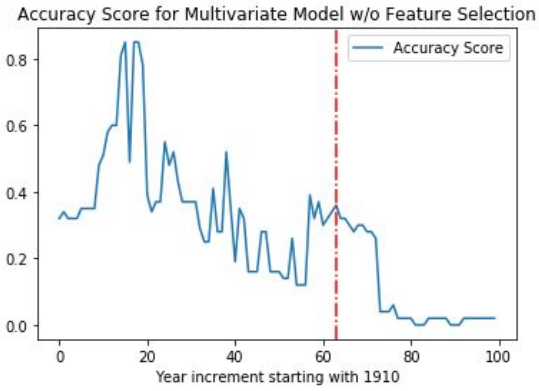

Fig. 11: Accuracy Score of predictions by Multivariate Model w/o Feature selection

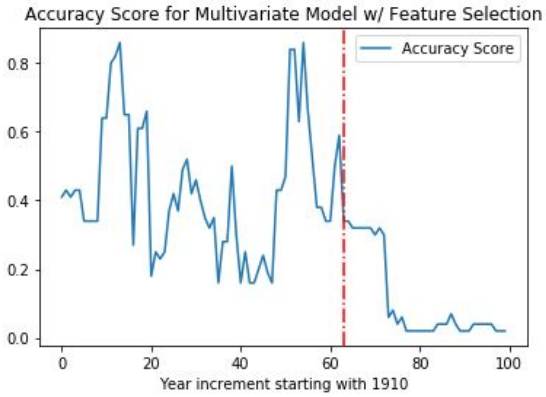

Fig. 12: Accuracy Score of predictions by Multivariate Model w/ Feature selection

Figure 11 and 12 showed a comparison on accuracy score over 105 years between model train by original dataset and reduced dataset, and we show the comparison of average accuracy score over the train and test set for both model in TABLE IV. They demonstrate that even though feature selection improved the performance significantly, it does not give the same significant improvement on accuracy score but only a slight one. This might due to the fact that the reduced feature dimension is still too high for the model to fit with such limited amount of samples.

Dataset	#Features	AAS(Training)	AAS(Test)
Original	93805	0.37	0.09
Reduced	35049	0.44	0.12

TABLE IV: Comparison between Average Accuracy Score

D. Independent Univariate LSTM Model Set

At the extended period of this project, we modified the existing model and developed the independent univariate LSTM model set in order to train the dataset without the problem of too high dimensionality. However, this model set requires extensive amount of training time since we need to train a separate model for each unique features. A trail testing shows that training time of the model with with simplest setting(with very limited amount of epochs and only 1 years back data) take 15 hours to train. We also start a formal training with higher number of epochs and higher count of years back. However, by the time this report is submitted, the training is still in progress thus we can not show the true performance of this model in this report. Instead we show the predicted density for the Model Set using the simplest setting just for reference.

VII. Conclusion

This project analyze the USA-Baby-Name dataset and incorporated various ways to apply LSTM(Long Short Term Memory) based RNN(Recurrent Neural Network) in order to predict the top-10 most popular baby name given name frequency of previous years. During the process of developing the model, we encountered the dimensionality problem where the number of features is much higher than the number of observations($p >> n$ problem). We tried various way to

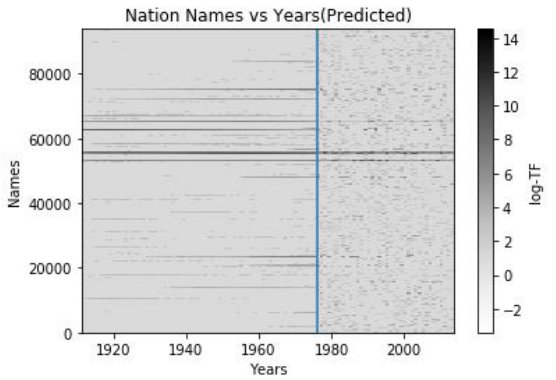

Fig. 13: Predicted Density for Indepent Model Set(Simpliest setting)

alleviate the impact of the dimensionality problem such as reducing the feature dimension by performing feature selection, developed a technique called the independent model set. The experiment showed that the feature selection process provided a positive effect to both the training performance and resulting accuracy. However, since the number of sample data is limited, the model is some times overfitting such that it does not perform well on the test set. If possible, we would like to search for an more defined dataset, such as Name Counts per month instead of per years so that we could use more input samples to train the model to avoid overfitting. Also, because the existence of time limitation, we were not able to finish collecting experiment result for independent univariate model set. We are looking forward to see the result of the univariate model and compare it with the multivariate one.

Through the experiments, we know that machine learning models like LSTM requires a lot of sample observations in order to train the model precisely. Even though in this project we did not have enough samples, we were thrilled to find ways around in order to solve a p¿¿n problem and it seems to be performing well. Furthermore, we realized that hardware limitation is a very realistic problem: even with a small dataset like USA-Baby-Name. We encountered many OOM(Out-of-Memory) errors when trying to improve the model. This inspired us to think of the possibilities of training model distributively in parallel. If we had more time, we would like to investigate in this area as well.

Acknowledgement

We thank kreas[5], scipy[11], pandas[9], TensorFlow[6] and scikitl-earn[10] for providing us a perfect set of machine learning related API so that we could frame the data and train the model easily.

We thank J. Brownlee[1] [7] for inspiring us how to process time series data through RNN network.

We thank Wikipedia editors for providing nice definition and explanation of RNN[3] modeland related problem[4], F. Deloche for sharing the wonderful graphic presentation[8] of RNN.

Lastly, we thank Data.gov[2] for providing us the original dataset on kaggle.com.

Thank again for all of the provider mention above, this project would not be possible without their contribution.

References

[1] J. Brownlee, "How to convert a time series to a supervised learning problem in python," Mar 2017. [Online]. Available: https://machinelearningmastery.com/convert-time-series-supervised-learning-problem-python/

[2] Data.gov, "Usa name data," Apr 2018. [Online]. Available: https://www.kaggle.com/datagov/usa-names

[3] "Recurrent neural network," Dec 2018. [Online]. Available: https://en.wikipedia.org/wiki/Recurrent_neural_networkLSTM

[4] "Vanishing gradient problem," Dec 2018. [Online]. Available: https://en.wikipedia.org/wiki/Vanishing_gradient_problem

[5] F. Chollet *et al.*, "Keras," https://keras.io, 2015.

[6] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[7] J. Brownlee, "Multivariate time series forecasting with lstms in keras," Sep 2018. [Online]. Available: https://machinelearningmastery.com/multivariate-time-series-forecasting-lstms-keras/

[8] F. Deloche, "Recurrent neural network," Dec 2018. [Online]. Available: https://en.wikipedia.org/wiki/Recurrent_neural_network/media/File:Long_Short-Term_Memory.svg

[9] "Python data analysis library." [Online]. Available: https://pandas.pydata.org/

[10] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[11] "Scipy.org." [Online]. Available: https://www.scipy.org/