ECE417 Final Project Report Team A

Adam Argiris
aradam@e-ce.uth.gr

Koffas George
gkoffas@e-ce.uth.gr

Chatzistefanidis Ilias
ichatsist@e-ce.uth.gr

Abstract

Global warming is a real and imminent threat for humanity. Its consequences are serious and have the potential to bring tremendous unrest in the world. Over the past 50 years, the average global temperature has increased at the fastest rate in recorded history. Additionally, experts see the trend is accelerating, and unless action is taken, the worst will come. So how can we use Machine Learning to tackle this problem?

Studies have suggested that Machine Learning can greatly contribute to this cause in many fields, from electricity systems, to CO_2 removal and even climate prediction and social geoengineering. In this report we will explore the best Machine Learning solutions for the Time Series Forecasting problem of Average Land Temperature measurements.

Keywords: *global warming, machine learning, time series*

1. Introduction

Global warming is the ongoing rise of the average temperature of the Earth's climate system and has been demonstrated by direct temperature measurements and by measurements of various effects of the warming.

It is a major aspect of climate change which, in addition to rising global surface temperatures, also includes its effects, such as changes in precipitation. While there have been prehistoric periods of global warming, observed changes since the mid-20th century have been unprecedented in rate and scale.[1]

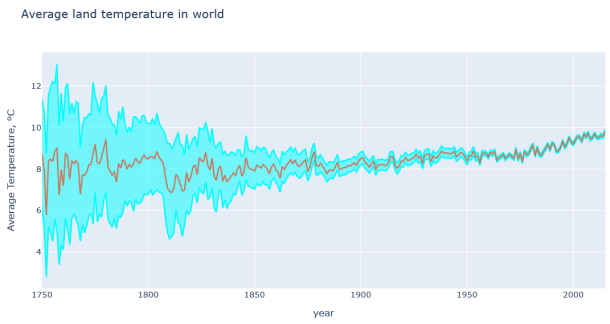

Figure 1: Average Land Temperature over 260 years. We can see an obvious increase on the slope of the graph.

A number of studies have indicated that global warming and climate change is responsible for a series of adverse effects, observed over a long period of time. Such effects include raise in ocean and sea levels[2][3][4], increased frequency of natural disasters such as hurricanes, tornadoes and floods[5], and even the spread of diseases.[6][7]

Over the years, a new field of research trying to tackle global warming has emerged: Machine Learning and its applications can be a powerful tool in reducing greenhouse gas emissions[8][9], and forecasting global temperature values.[10]

On this report, we will explore the problem of time-series forecasting on a dataset of temperature measurements and use a variety of Machine Learning models, to predict future temperatures and evaluate how accurate each model is for this specific problem. We will use a wide range of methods, from **Linear Regression** and **SVMs** to **LSTM** Neural Networks and **AutoML** algorithms. The rest of the report is structured into the following sections:

Related Work In this section we will present any publicized work related to our own.

Proposed Methods In this section we will present all the methods we used for our research accompanied by their mathematical and/or visual representations.

Experiments This section will cover the hyper-parameters used to create each model, the datasets and the software/hardware used.

Results and Discussion In this section we will show our experimental results, and discuss each model's flaws as well as strengths in said experiments.

Conclusions Lastly, in this section we will extract our conclusions over the experimental results and decide which models are best suited for the given problem.

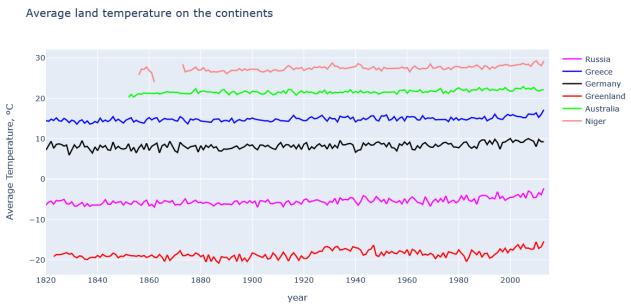

Figure 2: Average Land Temperature on the continents. We can see an obvious increase in average temperature in the colder regions of the world.

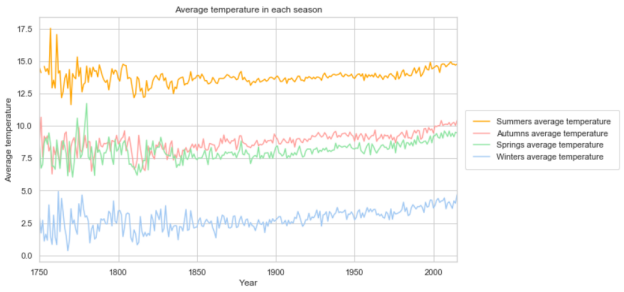

Figure 3: Average Seasonal Land Temperature. We can see an obvious increase in average temperature for each of the four seasons.

2. Related Work

Some related work to ours is listed below:

- Ye et al.[11] used Seasonal ARIMA (SARIMA) to make temperature predictions on the monthly global merged land-air-sea temperature data provided by the National Climate Data Center (NCDC) at the U.S. National Oceanic and Atmospheric Administration (NOAA). The model they used to deal with the time series of absolute temperatures T was given by:

$$T = L + C + E$$

  where L and C could be represented by analytical models, thus deterministic components, and E could be considered as the realization of a SARIMA process, thus the stochastic component of T.

- Suryono et al.[12] developed a web-based application for predictions using internet connection. For the predictions, a Fuzzy Time Series Algorithm induced by Markov transition matrix was used and pre-installed in their web-server.

3. Proposed Methods

3.1. Ridge Regression

Tikhonov regularization, named for Andrey Tikhonov, is a method of regularization of ill-posed problems. Also known as ridge regression, it is particularly useful to mitigate the problem of multicollinearity in linear regression, which commonly occurs in models with large numbers of parameters. In general, the method provides improved efficiency in parameter estimation problems in exchange for a tolerable amount of bias.

In the simplest case, the problem of a near-singular moment matrix $(\mathbf{X}^\mathsf{T}\mathbf{X})$ is alleviated by adding positive elements to the diagonals. The approach can be conceptualized by posing a constraint $\sum \beta_i^2 = c$ to the least squares problem, such that:

$$\min_{\beta} (\mathbf{y} - \mathbf{X}\beta)^\mathsf{T}(\mathbf{y} - \mathbf{X}\beta) + \lambda(\beta^\mathsf{T}\beta - c)$$

where λ is the Lagrange multiplier of the constraint. The minimizer of the problem is the simple ridge estimator:

$$\hat{\beta}_R = (\mathbf{X}^\mathsf{T}\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^\mathsf{T}\mathbf{y}$$

where $\mathbf{I}$ is the identity matrix and the ridge parameter λ serves as the positive constant shifting the diagonals, thereby decreasing the condition number of the moment matrix.

We will apply ridge regression in order to study the global temperatures' behaviour.

3.2. Linear Regression

A linear regression model assumes that the relationship between the dependent variable y and the p-vector of regressors x is linear.

With predictor variables x, the prediction of y is expressed by the following equation:

$$y = b_0 + b_1 * x$$

The regression beta coefficients measure the association between each predictor variable and the outcome. b_1 can be interpreted as the average effect on y of a one unit increase in x, holding b_0 fixed.

3.3. Support Vector Machine (SVM)

Support Vector Machines are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. Given a set of

training samples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new test samples to one category or the other, making it a non-probabilistic binary linear classifier.

The basic mathematics behind the SVM is however less familiar to most of us. It relies on the definition of hyperplanes and the definition of a margin which separates classes (in case of classification problems) of variables. It is also used for regression problems.

With SVMs we distinguish between hard margin and soft margins. The latter introduces a so-called softening parameter. We distinguish also between linear and non-linear approaches. The latter are the most frequent ones since it is rather unlikely that we can separate classes easily by say straight lines.

The Support Vector Regression (SVR) uses the same principles as the SVM for classification, with only a few minor differences. First of all, because output is a real number it becomes very difficult to predict the information at hand, which has infinite possibilities. In the case of regression, a margin of tolerance (epsilon) is set in approximation to the SVM which would have already requested from the problem. But besides this fact, there is also a more complicated reason, the algorithm is more complicated therefore to be taken in consideration. However, the main idea is always the same: to minimize error, individualizing the hyperplane which maximizes the margin, keeping in mind that part of the error is tolerated.

3.4. Kernel SVM

SVM algorithms use a set of mathematical functions that are defined as the kernel. The function of kernel is to take data as input and transform it into the required form. Different SVM algorithms use different types of kernel functions. These functions can be different types. Some of them are shown in Figure 4.

For example linear, nonlinear, polynomial, radial basis function (RBF) and sigmoid

introduce Kernel functions for sequence data, graphs, text, images, as well as vectors. The most used type of kernel function is RBF (gaussian) and it is the default one, because it has localized and finite response along the entire x-axis. The kernel functions return the inner product between two points in a suitable feature space. Thus by defining a notion of similarity, with little computational cost even in very high-dimensional spaces.

$k(\mathbf{x_i}, \mathbf{x_j}) = (\mathbf{x_i} \cdot \mathbf{x_j} + 1)^d$

Polynomial kernel equation

$k(x, y) = \tanh(\alpha x^T y + c)$

Sigmoid kernel equation

$k(x, y) = \exp\left(-\frac{\|x - y\|^2}{2\sigma^2}\right)$

Gaussian kernel equation

Figure 4: SVM Kernels

3.5. Seasonal Auto Regressive Integrated Moving Average (SARIMA)

3.5.1 Non seasonal ARIMA

We can split the Arima term into three terms; AR, I, MA:

- **AR(p)** stands for autoregressive model, the 'p' parameter is an integer that confirms how many lagged series are going to be used to forecast periods ahead, example:
  - The average temperature of yesterday has a high correlation with the temperature of today, so we will use AR(1) parameter to forecast future temperatures.
  - The formula for the AR(p) model is:

    $$\hat{y}_t = \mu + \theta_1 Y_{t-1} + ... + \theta_p Y_{t-p}$$

    where μ is the constant term, p is the periods to be used in the regression and θ is the parameter fitted to the data.
- **I(d)** is the differencing part, the 'd' parameter tells how many differencing orders are going to be used, it tries to make the series stationary, example:
  - Yesterday I sold 10 items of a product, today I sold 14, the "I" in this case is just the first difference, which is +4, if you are using logarithm base this difference is equivalent to percentual difference.
  - If d = 1:

    $$y_t = Y_t - Y_{t-1}$$

    where y_t is the differenced series and $Y_{t-period}$ is the original series
  - If d = 2:

    $$y_t = (Y_t - Y_{t-1}) - (Y_{t-1} - Y_{t-2}) = Y_t - 2Y_{t-1} + Y_{t-2}$$

  - Note that the second difference is a change-in-change, which is a measure of the local "acceleration" rather than trend.

- **MA(q)** stands for moving average model, the ‘q‘ is the number of lagged forecast errors terms in the prediction equation, example:
  - It’s strange, but this MA term takes a percentage of the errors between the predicted value against the real. It assumes that the past errors are going to be similar in future events.
  - The formula for the MA(p) model is:

    $$\hat{y}_t = \mu - \Theta_1 e_{t-1} + ... + \Theta_q e_{t-q}$$

    where μ is the constant term, q is the period to be used on the e term and Θ is the parameter fitted to the errors
  - The error equation is

    $$e_t = Y_{t-1} - \hat{y}_{t-1}$$

3.5.2 Seasonal ARIMA

The p, d, q parameters are capitalized to differ from the non seasonal parameters.

- **SAR(P)** is the seasonal autoregression of the series.
  - The formula for the SAR(P) model is:

    $$\hat{y}_t = \mu + \theta_1 Y_{t-s}$$

    where P is quantity of autoregression terms to be added, usually no more than 1 term, s is how many periods ago to be used as base and θ is the parameter fitted to the data.
  - Usually when the subject is weather forecasting, 12 months ago have some information to contribute to the current period.
  - Setting P=1 (i.e., SAR(1)) adds a multiple of Y_{t-s} to the forecast for y_t
- **I(D)** the seasonal difference must be used when you have a strong and stable pattern.
  - If d = 0 and D = 1:

    $$y_t = Y_t - Y_{t-s}$$

    where y_t is the differenced series and Y_{t-s} is the original seasonal lag.
  - If d =1 and D = 1: $y_t = (Y_t - Y_{t-1}) - (Y_{t-s} - Y_{t-s-1}) = Y_t - Y_{t-1} - Y_{t-s} + Y_{t-s-1}$
  - D should never be more than 1, and d+D should never be more than 2. Also, if d+D =2, the constant term should be suppressed.
- **SMA(Q)**
  - Setting Q=1 (i.e., SMA(1)) adds a multiple of error e_{t-s} to the forecast for y_t
- **S** It’s the seasonal period where you are going to calculate the P, D, Q terms. If there is a 52 week seasonal correlation this is the number to be used on the ’S’ parameter

3.5.3 Trend

We will use **SARIMAX** to create a forecast, the following terms are a definition to the trend:

- ’n’ when there is no trend to be used (default).
- ‘c’ indicates a constant (i.e. a degree zero component of the trend polynomial)
- ‘t’ indicates a linear trend with time
- ‘ct’ is both trend and constant.
- Can also be specified as an iterable defining the polynomial as in numpy.poly1d, where [1,1,0,1] would denote $a + bt + ct^3$

3.6. Long short-term memory (LSTM)

Long short-term memory (LSTM) is an artificial recurrent neural network (RNN) architecture used in the field of deep learning. Unlike standard feedforward neural networks, LSTM has feedback connections. It can not only process single data points such as images, but also entire sequences of data such as speech or video. For example, LSTM is applicable to tasks such as unsegmented, connected handwriting recognition, speech recognition and anomaly detection in network traffic or IDS’s (intrusion detection systems).

There are several architectures of LSTM units. A common architecture is composed of a cell - the memory part of the LSTM unit - and three ”regulators”, usually called gates, of the flow of information inside the LSTM unit: an input gate, an output gate and a forget gate. Some variations of the LSTM unit do not have one or more of these gates or maybe have other gates. For example, gated recurrent units (GRUs) do not have an output gate.

Intuitively, the cell is responsible for keeping track of the dependencies between the elements in the input sequence. The input gate controls the extent to which a new value flows into the cell, the forget gate controls the extent to which a value remains in the cell and the output gate controls the extent to which the value in the cell is used to compute the output activation of the LSTM unit. The activation function of the LSTM gates is often the logistic sigmoid function.

There are connections into and out of the LSTM gates, a few of which are recurrent. The weights of these connections, which need to be learned during training, determine how the gates operate.

LSTM networks are well-suited to classifying, processing and making predictions based on time series data, since there can be lags of unknown duration between important events in a time series. LSTMs were developed to deal with the vanishing gradient problem that can be encountered when training traditional RNNs. Relative insensitivity to gap length is an advantage of LSTM over RNNs, hidden Markov models and other sequence learning methods in numerous applications.

3.7. XGBoost

XGBoost is an open-source software library which provides a gradient boosting framework for C++, Java, Python, R, Julia, Perl, and Scala. It works on Linux, Windows, and macOS. From the project description, it aims to provide a "Scalable, Portable and Distributed Gradient Boosting (GBM, GBRT, GBDT) Library". It runs on a single machine, as well as the distributed processing frameworks Apache Hadoop, Apache Spark, and Apache Flink. It has gained much popularity and attention recently as the algorithm of choice for many winning teams of machine learning competitions.

XGBoost is used for supervised learning problems, where we use the training data (with multiple features) x_i to predict a target variable y_i.

XGBoost saves the trees, some model parameters like number of input columns in trained trees, and the objective function, which combined to represent the concept of "model" in XGBoost. As for why are it is saving the objective as part of model, that's because objective controls transformation of global bias (called base_score in XGBoost). Users can share this model with others for prediction, evaluation or continue the training with a different set of hyperparameters etc. However, this is not the end of story. There are cases where it needs to save something more than just the model itself. For example, in distrbuted training, XGBoost performs checkpointing operation. Or for some reasons, your favorite distributed computing framework decide to copy the model from one worker to another and continue the training in there. In such cases, the serialisation output is required to contain enougth information to continue previous training without user providing any parameters again. It considers such scenario as memory snapshot (or memory based serialisation method) and distinguish it with normal model IO operation.

3.8. Automated machine learning (AutoML)

AutoML is the process of automating the process of applying machine learning to real-world problems. AutoML covers the complete pipeline from the raw dataset to the deployable machine learning model. AutoML was proposed as an artificial intelligence-based solution to the ever-growing challenge of applying machine learning. The high degree of automation in AutoML allows non-experts to make use of machine learning models and techniques without requiring to become an expert in this field first.

Automating the process of applying machine learning end-to-end additionally offers the advantages of producing simpler solutions, faster creation of those solutions, and models that often outperform hand-designed models.

The libary we use is H20. Although H2O has made it easy for non-experts to experiment with machine learning, there is still a fair bit of knowledge and background in data science that is required to produce high-performing machine learning models. Deep Neural Networks in particular are notoriously difficult for a non-expert to tune properly. In order for machine learning software to truly be accessible to non-experts, we have designed an easy-to-use interface which automates the process of training a large selection of candidate models. H2O's AutoML can also be a helpful tool for the advanced user, by providing a simple wrapper function that performs a large number of modeling-related tasks that would typically require many lines of code, and by freeing up their time to focus on other aspects of the data science pipeline tasks such as data-preprocessing, feature engineering and model deployment.

H2O's AutoML can be used for automating the machine learning workflow, which includes automatic training and tuning of many models within a user-specified time-limit. Stacked Ensembles – one based on all previously trained models, another one on the best model of each family – will be automatically trained on collections of individual models to produce highly predictive ensemble models which, in most cases, will be the top performing models in the AutoML Leaderboard.

4. Experiments

4.1. Dataset

We used the following list of datasets from Kaggle:

- *GlobalLandTemperaturesByCity.csv*
- *GlobalLandTemperaturesByCountry.csv*
- *GlobalTemperatures.csv*

The label column for each one is **AverageTemperature** and **LandAverageTemperature** (for GlobalTemperatures.csv).

4.2. Software

All code was written in Python 3.7 version, using the Anaconda 3.0 suite and specifically the Jupyter Notebook tool. A list of libraries used for the training and testing of the models, and visualization of the results follows:

- numpy, v1.18.1 (`https://numpy.org/doc/1.18/`)
- pandas, v1.0.3 (`https://pandas.pydata.org/docs/`)
- matplotlib, v3.1.3 (`https://matplotlib.org/3.2.1/contents.html`)
- plotly, v4.7.1 (`https://plotly.com/python/`)
- seaborn, v0.10.0 (`https://seaborn.pydata.org/api.html`)
- statsmodels, v0.11.1 (`https://www.statsmodels.org/stable/api.html`
- scikit-learn, v0.22.2.post1 (`https://devdocs.io/scikit_learn/`)
- chart_studio, v.1.1.0 (`https://plotly.com/chart-studio/`)
- colorlover, v0.3.0 (`https://github.com/plotly/colorlover`)
- keras, v2.3.1 (`https://keras.io/api/`)
- xgboost, v1.0.2 (`https://xgboost.readthedocs.io/en/latest/`)
- imageio, v2.8.0 (`https://imageio.readthedocs.io/en/stable/`)
- h2o, v3.30.0.3 (`https://docs.h2o.ai/h2o/latest-stable/h2o-docs/automl.html`)

4.3. Preprocessing and Training

Table 1 shows the hyper-parameter values of each model. We will examine each model on our datasets, and the results we extract will be further discussed in Section 5.

Specifically for the LSTM model, a visualization of the resulting network can be seen in Figure 5.

During preprocessing, categorical columns were mostly dropped, so we ended up working with a univariate dataset most of the time. Our approach to some of those datasets was to introduce a time-shifted column as one of our features (e.g. LSTM, AutoML), but we also used a type of One-hot Encoding for others (e.g. XGBoost).

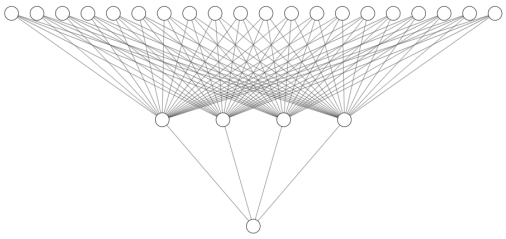

Figure 5: LSTM network architecture. From top to down: *Input Layer*, *Hidden LSTM Layer*, *Output Layer*.

5. Results and Discussion

Every method shows a different behaviour when we try to apply it at our data.

5.1. Ridge Regression

Ridge Regression helps us understand the general trend in temperature data through time and not the precise values. By changing alpha and the degree of the polynomial - when it comes to polynomial ridge regression - we can have a better picture of this trend.

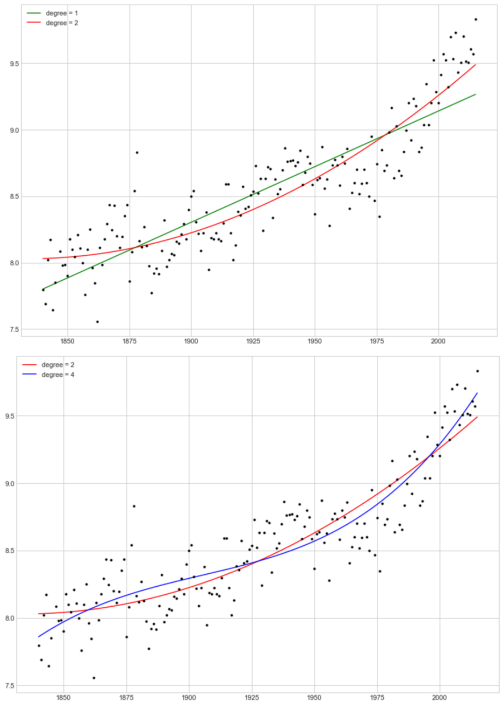

Figure 6: Ridge Regression with 1st, 2nd and 4th polynomial's degree applied on global temperatures *(from 1830-2012)*.

In Figure 6, we can clearly observe that there is an uptrend in global temperatures through time. Specifically, the poly-

Model	Libraries	Parameters
Lin. Ridge Reg.	sklearn	$a \in \{0, 1, 0.5e5, 0.5e6\}, *$
Poly. Ridge Reg.	sklearn	$deg \in \{1, 2, 4\}, a \in \{0, 1, 50\}, *$
Lin. Regression	statsmodels	$*$
SVM	sklearn	$C = 1e3, \gamma = 1, \epsilon = 0.01, *$
Poly. Kernel SVM	sklearn	$deg = 3, C = 1e{-}5, \gamma = 1e{-}4, \epsilon = 0.1, *$
Gaussian Kernel SVM	sklearn	$C = 1.12, \gamma = 0.33, \epsilon = 1e{-}4, *$
Sig. Kernel SVM	sklearn	$C = 1e3, \gamma = 1, \epsilon = 0.01, *$
Lin. Kernel SVM	sklearn	$C = 1, \gamma = 1, \epsilon = 0.1, *$
SARIMA	statsmodels	$(p, d, q) = (3, 0, 0), (P, D, Q, S) = (0, 1, 1, 12), \text{trend} = c, *$
LSTM	keras	$\text{nodes} = 4, \text{loss} = MSE, \text{optimizer} = Adam, *$
XGBoost	sklearn	$\text{n\_estimators} = 1000, \text{early\_stopping\_rounds} = 50, *$
AutoML	H2O	$\text{max\_runtime\_secs} = 60, seed = 1, *$

Table 1: Model hyper-parameters. For entries with an asterisk symbol, rest of values set to default.

nomial of 2nd degree fits better to the data. This means, that not only global temperature is increased, but also in an exponential way.

5.2. Linear Regression

The correlation in our data are so complicated which leads to failure or low accuracy of Linear Regression. The only information that we can withdraw from this analysis is that there is an uptrend in our data, as it appears in Figure 7.

5.3. SVM and Kernel SVM

Unfortunately, kernel SVM - in specific sigmoid, polynomial and linear kernels - did not have the results we expected. They either have the same behaviour as Linear Regression or completely fail to fit data, as we can see in Figure 8.

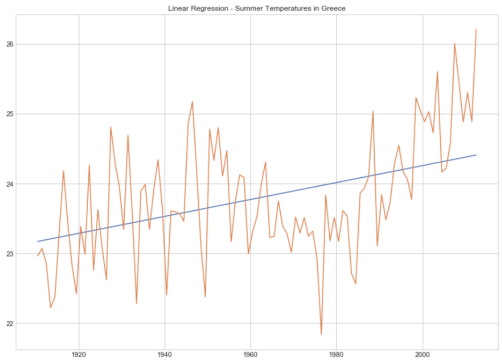

Figure 7: Linear Regression on Athens summer temperatures from 1900 to 2012.

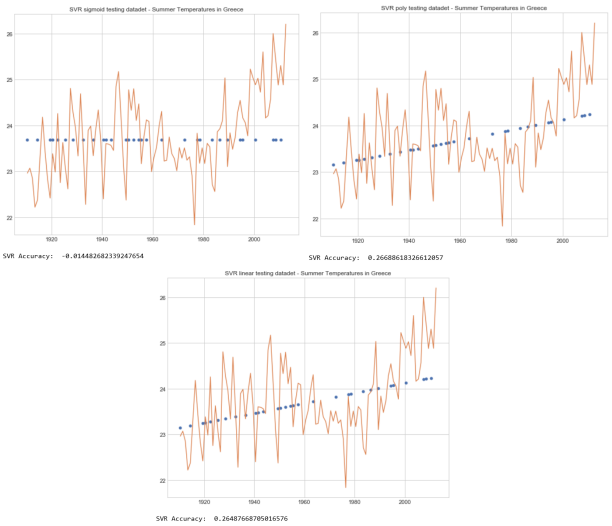

Figure 8: Failure and low accuracy of SVM sigmoid, polynomial and linear kernel.

On the other side, RBF - default gaussian - kernel gives better results. As we can see in Table 2, it achieves - with the initial parameters - nearly 100% training accuracy and 35% testing accuracy. However, this big difference between training and testing accuracy indicates overfitting of the model.

Thus, in order to reduce overfitting, we optimized our model by selecting new parameters:

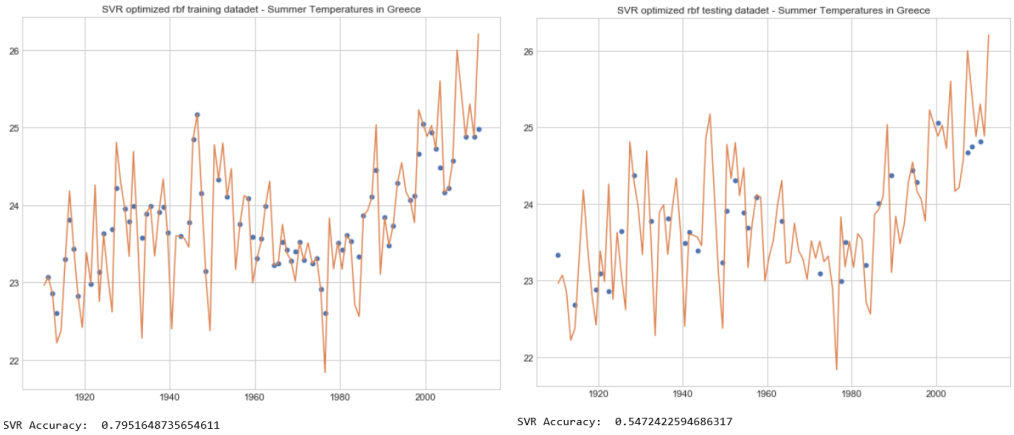

Figure 9: SVM with RBF Kernel performance on train and test datasets after accounting for overfitting.

- $C = 1.12$,
- $\gamma = 0.33$,
- $\epsilon = 0.00001$

In this way, we increased prediction accuracy - as we can see in Figure 9. and in Table 2. - by achieving 80% training and 55% testing accuracy.

Parameters	Train Accuracy	Test Accuracy
Initial	0.9998	0.3455
Optimized	0.7952	0.5472

Table 2: SVM RBF: Fixing overfitting

5.4. LSTM

Neural Networks are known to perform really well in a wide range of problems, including TSF problems. However, a common difficulty that arises is the problem of data leakage in univariate datasets, especially when using Recurrent Neural Network architectures such as LSTM nodes.

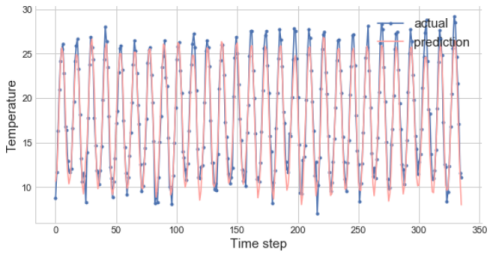

Figure 10: LSTM predictions over the test dataset *(Athens temperatures up to 2012)*.

After taking these thoughts into consideration and accounting for the data leakage problems, we came up with a really good solution. Our LSTM model performed extremely well, both in terms of time performance and accuracy.

The resulting predictions can be seen in Figure 10.

5.5. XGBoost

XGBoost, as part of decision tree-based algorithms, is well accepted to perform efficiently and with great accuracy when dealing with multivariate datasets. However, decision trees are subject to overfitting and XGBoost is no exception.

Our approach deals with the mentioned above problems by applying One-hot Enconding. In this way, we achieve highly appreciable results as can be seen in Figure 11.

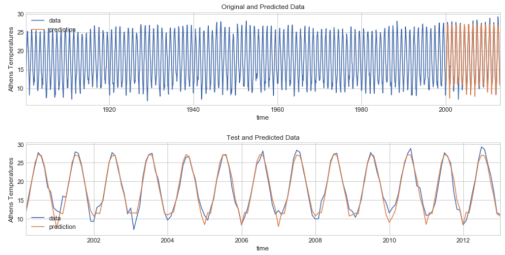

Figure 11: XGBoost predictions for Athens temperatures from 2000 to 2012.

5.6. AutoML

AutoML presented us with the opportunity to use state-of-the-art algorithms for machine learning in an easy to code and understand way. Its performance was high and up to standards, while the results was sufficiently accurate. This can be seen in Figure 12.

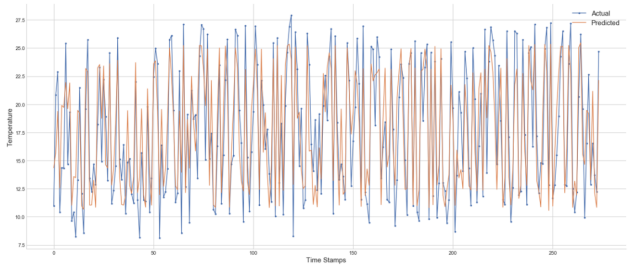

Figure 12: AutoML winning model predictions for Athens temperatures up to 2012.

5.7. SARIMA

SARIMA performed the best out of the other models in terms of prediction accuracy. In the beginning, we dealt with the seasonal correlation of temperature values by implementing a stationary series; constant mean, variance and autocorrelation.

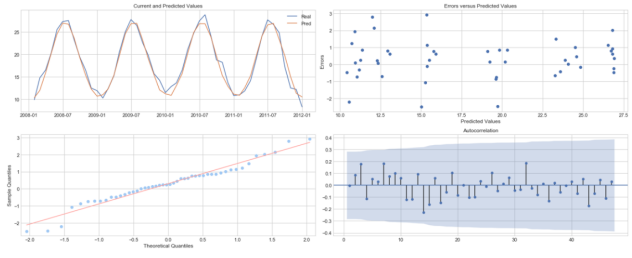

Figure 13: SARIMA stationarity plots (*Athens temperatures from 2008-2012*).

We managed to achieve these characteristics, as we can see in Figure 13.

- The Error vs Predicted values has a linear distribution (the errors are between -3 and $+3$ while the temperature increases).
- The QQ Plot shows a normal pattern with some little outliers and,
- The autocorrelation plot shows a positive spike over the confidence interval just above the third lag, but we believe that there is no need for more changes.

Finally, the predictions from the resulting model are included in Figure 14.

6. Conclusions

Based on our results, we can conclude that the best performing models for Time Series Forecasting for our datasets are SARIMA, XGBoost, LSTM Neural Network and AutoML.

Depending on the intricacies of your specific application, different models are more useful than others. For instance, if your application is dependent on time performance AutoML is best suited. On the other hand, if it demands high accuracy LSTM and SARIMA solutions should be preferred.

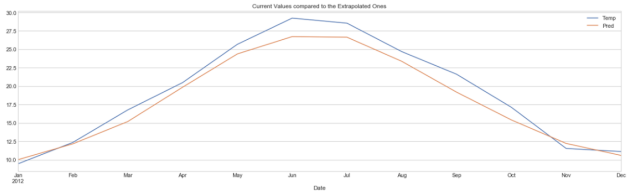

Figure 14: SARIMA predictions for Athens temperatures for the year 2012.

Future Work In the future, we could extend our work and observations by:

- examining more models, such as Facebook's Prophet,
- fit our models on different datasets that are related to the Global Warming TSF problem, in order to give a more conclusive answer.

7. Acknowledgements

1. Kaggle's **"Climate Change: Earth Surface Temperature Data"** dataset and the related Kernels developed for the dataset: used for the purposes of this report.
2. Alex Lenail's **NN SVG**: used to visualize our LSTM network architecture.
3. The **IEEEXplore** and **arXiv** libraries: used to find any other relevant research.

8. Contributions

Each member of our team contributed equally in order this project to come to fruition.

References

[1] F. Einaudi, "Climate change and 50 years of nasa observations," in *2013 International Conference on Electromagnetics in Advanced Applications (ICEAA)*, pp. 1548–1549, 2013.

[2] S. Jin, L. Zhang, and G. Feng, "Earth's surface fluid variations and deformations from gps and grace in global warming," 2011.

[3] L. Hinnov, K. N. Ramamurthy, H. Song, M. Banavar, and L. Spanias, "Interactive tools for global sustainability and earth systems: Sea level change and temperature," in *2013 IEEE Frontiers in Education Conference (FIE)*, pp. 1965–1970, 2013.

[4] W. G. S. L. B. Group, "Global sea-level budget 1993–present," *Earth System Science Data*, vol. 10, no. 3, pp. 1551–1590, 2018.

[5] J. Chen and Y. Gao and L. Wen, "A study of the relationship between global warming and hurricane activity based on time series," in *2018 International Conference on Robots Intelligent System (ICRIS)*, pp. 460–463, 2018.

[6] B. Andrick, B. Clark, K. Nygaard, A. Logar, M. Penaloza, and R. Welch, "Infectious disease and climate change: detecting contributing factors and predicting future outbreaks," in *IGARSS'97. 1997 IEEE International Geoscience and Remote Sensing Symposium Proceedings. Remote Sensing - A Scientific Vision for Sustainable Development*, vol. 4, pp. 1947–1949 vol.4, 1997.

[7] V. Petraşcu, G. Czibula, D. Sîrbu, M. Popa, and D. Curşeu, "Identifying the impact of global warming on population using a clustering based approach," in *2010 IEEE International Conference on Automation, Quality and Testing, Robotics (AQTR)*, vol. 2, pp. 1–6, 2010.

[8] D. Rolnick, P. L. Donti, L. H. Kaack, K. Kochanski, A. Lacoste, K. Sankaran, A. S. Ross, N. Milojevic-Dupont, N. Jaques, A. Waldman-Brown, A. Luccioni, T. Maharaj, E. D. Sherwin, S. K. Mukkavilli, K. P. Körding, C. Gomes, A. Y. Ng, D. Hassabis, J. C. Platt, F. Creutzig, J. Chayes, and Y. Bengio, "Tackling climate change with machine learning," *CoRR*, vol. abs/1906.05433, 2019.

[9] E. GARİP and A. B. OKTAY, "Forecasting co2 emission with machine learning methods," in *2018 International Conference on Artificial Intelligence and Data Processing (IDAP)*, pp. 1–4, 2018.

[10] Himika, S. Kaur, and S. Randhawa, "Global land temperature prediction by machine learning combo approach," in *2018 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, pp. 1–8, 2018.

[11] L. Ye, G. Yang, V. Ranst, Eric, and H. Tang, "Time-series modeling and prediction of global monthly absolute temperature for environmental decision making," *Advances in Atmospheric Sciences*, vol. 30, pp. 382–396, 03 2013.

[12] S. Suryono, R. Saputra, B. Surarso, and H. Sukri, "Web-based fuzzy time series for environmental temperature and relative humidity prediction," in *2017 IEEE International Conference on Communication, Networks and Satellite (Comnetsat)*, pp. 36–41, 2017.