Final Report

Ellie Strande, Nathan Nguyen, Nayeli Castro, Liora Mayats Alpay

Introduction

The goal of our machine learning algorithm was to improve longer-range weather forecasts of temperature and precipitation by blending physics-based forecasts with machine learning. The dataset provided contains meteorological data capable of improving sub-seasonal forecasts for weather and climate conditions, as well as predicting natural disasters in advance. Having improved long-term forecasts is important as it will help communities and industries adapt to the challenges brought on by climate change, including providing people the opportunity to prepare for extreme weather events. Further, the ability to generate long-term climate predictions will be helpful for research involving ecosystems, wildlife, human societies, and economies.

Climate change refers to the long-term changes in the Earth's climate system's average temperature and weather patterns. It is caused by human activities, primarily the burning of fossil fuels, which releases greenhouse gases into the atmosphere and traps heat from the sun, leading to global warming. The effects of climate change include rising sea levels, more intense heat waves, increased frequency and severity of extreme weather events, and changes in precipitation patterns. Climate change is also having a significant impact on ecosystems and wildlife, as well as human societies and economies.

By implementing a machine learning algorithm that can predict longer-range weather forecasts of temperature and precipitation, it is hoped that the adverse consequences of climate change will be detected and prepared for long in advance. With purely physics-based models dominating current-day weather forecasting, which is primarily sufficient for short-term predictions, there is a limited forecast horizon. This could be consequential when trying to adapt and prepare for adverse conditions brought on by climate change.

Data

About the Dataset

The dataset we were provided was quite extensive, featuring over 200 predictor variables. The data was sourced from a variety of locations and covered a wide range of categories, incliding Madden-Julian oscillation, pressure, and evaporation. We utilized variables such as sea level pressure (slp), MJO phase and amplitute, geopotential height at various millibar values, zonal wind, and weighted average forecasts for precipitation form the NMME model. Overall, the majority of predictor variables used in our model from this dataset were physics-based climate statistics measured in the past. These variables provided a comprehensive and diverse range of information to inform our model development.

Target Variable: contest-tmp2m-14d__tmp2m

The dataset we were working with was very large, which presented various challenges, especially when developing our model. Due to the high number of rows and predictor variables, the process was quite time-consuming and computationally intensive. Despite this, our group was able to find solutions that made the model development process more efficient. We decided to utilize XGBoost for our model as it is an efficient algorithm that is better suited to handle high-dimensional data. Using XGBoost allowed us to mitigate some of the computational challenges presented by the high dimensioanlity of the dataset.

Data Cleaning

To begin our analysis, we initiated a thorough data cleaning process. We began by eliminating any missing values by using the dropna() function. Next, we removed any categorical predictor variables that we would not be using in the model. Finally, we employed the technique of Principal Component Analysis (PCA) to optimize the performance fo the model by reducing the number of variables while preserving important information. We will be delving deeper into the specifics of this technique and its application in the modeling section.

```
#Drop missing values
trainData = trainData.dropna()
#Find which variables are strings/continuous (objects)
typesDF = pd.DataFrame(trainData.dtypes)
typesDF
#Create list of predictor variables
preds = list(trainData)
preds.remove('index')
preds.remove('lat')
```

```
preds.remove('startdate')
preds.remove('climateregions__climateregion')
preds.remove('mjo1d__phase')
preds.remove('mei__meirank')
preds.remove('mei__nip')
preds.remove('contest-tmp2m-14d__tmp2m')
X = trainData[preds]
y = trainData['contest-tmp2m-14d__tmp2m']
```

The data is linked below:

train data: https://www.kaggle.com/competitions/widsdatathon2023/data?select=train_data.csv

test data: https://www.kaggle.com/competitions/widsdatathon2023/data?select=test_data.csv

Modeling

To begin the model development process, we applied a Linear Regression model to the dataset. Linear regression models describe the relationship between a dependent variable and one or more independent variables, finding the line of best fit through the data points. In our initial attempt, we attemped to standardize the continuous variables by generating Z-scores. However, this method did not yield the desired results as it generated an unusually high mean squared error. After removing Z-scoring, the model performed substantially better, yielding a significantly lower mean squared error.

Simple Linear Regression Model

```
#Train test split
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size = 0.2,
                                                  random_state = 5)
#Standardize/Z score
z = StandardScaler()d
X_train[preds] = z.fit_transform(X_train[preds])
X_val[preds] = z.transform(X_val[preds])
#Create and fit model
```

```
lr = LinearRegression()
lr.fit(X_train, y_train)
#Predictions
y_pred = lr.predict(X_val)
#Print Training MSE
print('Train MSE: ', mean_squared_error(y_train, lr.predict(X_train)))
#Print Testing MSE
print('Test MSE: ', mean_squared_error(y_val, y_pred))
```

Although our linear regression model performed fairly well, we were not satisfied with the limited room for growth provided by this model. We also wanted to try a model that did not assume linearity, as linear regression models assume a linear relationship between the independent and dependent variables.

Next, we tried using Principal Component Analysis (PCA) to reduce the number of predictor variables in our model. With the large datset we were working with being very time-consuming and computationally expensive, we decided to try dimensionality reduction to reduce the number of variables while preserving important information. PCA reduces the dimensionality of large datasets by transforming data into a new set of linearly uncorrelated variables, called principal components. These principal components are combinations of the original variables that capture the most variance in the data. Consequentially, we found that our linear regression mean squared error was significantly higher when using the principal components in the model as opposed to the original variables.

XGBoost

With the goal of generating long-term predictions for temperature and precipitation, our group decided to use XGBoost, an open-source software library for gradient boosing on decision trees. XGBoost is good for time series data and the development of long-term models, which immediately made it a strong fit for this situation. Additionally, XGBoost provides the ability to make design decisions by tuning hyperparameters to determine the most optimally fit model. XGBoost is designed to be efficient and scalable, making it very successful in the past for Kaggle competitions. The amount of control XGBoost provides was appealing because we could continue to improve our model until we were satisfied with the results. XGBoost picks up on patterns and regularities in the data by automatically tuning thousands of learnable parameters, which is why the model demonstrates so much success in time-series scenarios.

Our XGBoost model with tuned hyperparameters generated a test mean squared error of 0.135 on the validation set, and a train mean squared error of 0.0656. This demonstrates that our model is slightly overfit.

XGBoost code:

```
#Split the data into seperate training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = random_state = 5)

#Define the xgboost regression model
model = XGBRegressor(n_estimators = 1000, max_depth = 10, learning_rate = 0.2,
                     colsample_bytree = 0.8, gamma = 0.15,
                     min_child_weight = 2, reg_alpha = 42, reg_lambda = 0.5805879363282948,
                     subsample = 0.8, scale_pos_weight = 1, nthread = 4, seed = 27)

#Train the model
model.fit(X_train, y_train)

#Predictions
y_pred = model.predict(X_test)

#Training MSE
print(mean_squared_error(y_train, model.predict(X_train)))

#Testing MSE
print(mean_squared_error(y_test, y_pred))

#PREDICTING ON TEST DATASET
finalPreds = model.predict(testData[preds])

#Create dataframe to store the results
results = pd.DataFrame(finalPreds, columns = ['contest-tmp2m-14d__tmp2m'])

results['index'] = testData['index']

results.to_csv('/Users/elliestrande/Desktop/WiDS/solution.csv', index = False)
```

While having so much control over the hyperparameters allows for a lot of success, it can also be a drawback due to the complexity of developing this type of model. With the large climate dataset we were working with, hyperparameter tuning methods such as GridSearch were not helpful as they took an extreme amount of time and often crashed due to a lack of memory space. Without the ability to use GridSearch and similar hyperparameter tuning methods, we were challenged with the task of learning new methods of hyperparameter tuning that better suited XGBoost.

Hyperparameter Tuning

The hyperparameters our model has include: - n_estimators - max_depth - learning_rate - colsample_bytree - gamma - min_child_weight - reg_alpha - reg_lambda - eta - subsample - scale_pos_weight - nthread - seed

To select the hyperparameters, we initially tried to utilize GridSearchCV and RandomSelectionCV, which are both built-in tools in many machine learning libraries. Both of these methods work by iterating over a predefined set of hyperparameter values and training a model on each combination, in order to find the combination that results in the best performance. Consequentally, in our case, we found that using GridSearchCV and RandomSelectionCV was not successful. When working with large datasets, these methods can become very time-consuming and resource-intensive. This is because each iteration of the algorithm requires training a new model, which can take a significant amount of time and computational resources. As the number of possible hyperparameter combination increases, the number of iterations required also increases, causing the problem to significantly exacerbate when using XGBoost. For our model, the process took an extreamly long time and often ended up crashing. This is likely due to the fact that the dataset was too large for these methods to handle efficiently. As a result, we had to look for alternative methods for selecting the optimal hyperparameters for our model.

Given the inefficiency of using GridSearch and RandomSelectionCV on our large dataset, we decided to resort to a more manual approach for selecting the optimal hyperparameters. This involved learning about the meaning and ideal value ranges for each parameter, and manually adjusting them in an attempt to improve model performence. To aid in the process, we employed a technique called Bayesian optimization, which uses a library called Hyperopt to find the best set of hyperparameters for our model. Hyperopt is a Python library that uses a search algorithm to find the hyperparameter values that minimize the loss function, which is a measure of how well the model is performing. By using Bayesian optimization with Hyperopt, we were able to more efficiently and effectively tune the hyperparameters of our model.

Baysian Optimization

```
#Initialize domain space for range of values
space={'max_depth': hp.quniform("max_depth", 3, 18, 1),
        'gamma': hp.uniform ('gamma', 1,9),
        'reg_alpha' : hp.quniform('reg_alpha', 40,180,1),
        'reg_lambda' : hp.uniform('reg_lambda', 0,1),
        'colsample_bytree' : hp.uniform('colsample_bytree', 0.5,1),
        'min_child_weight' : hp.quniform('min_child_weight', 0, 10, 1),
        'n_estimators': 180,
        'seed': 0,
```

```
    }

#Define objective function
def objective(space):
    clf=xgb.XGBRegressor(
                    n_estimators =space['n_estimators'], max_depth = int(space['max_depth'
                    reg_alpha = int(space['reg_alpha']),min_child_weight=int(space['min_ch
                    colsample_bytree=int(space['colsample_bytree']))

    evaluation = [( X_train, y_train), ( X_test, y_test)]

    clf.fit(X_train, y_train,
            eval_set=evaluation, eval_metric="rmse",
            early_stopping_rounds=10,verbose=False)


    pred = clf.predict(X_test)
    accuracy = r2_score(y_test, pred>0.5)
    print ("SCORE:", accuracy)
    return {'loss': -accuracy, 'status': STATUS_OK }

trials = Trials()

#Optimize
best_hyperparams = fmin(fn = objective,
                        space = space,
                        algo = tpe.suggest,
                        max_evals = 50,
                        trials = trials)

print("The best hyperparameters are : " , "\n")
print(best_hyperparams)
```

Improvements

One aspect of our model that we would like to further improve is the selection of hyperparameters. XGBoost is a powerful algorithm that offers a lot of flexibility when building and training a model. While the abundance of hyperparameters allows for fine-tuning and optimal performance, it can be challenging to navigate and optimize such a large number of options. Given more time, we would continue to work on tuning the hyperparameters to better suit

our dataset and improve the overall performance of the model.

Limitations

During the development of our model, we encountered several limitations in the data provided. One major issue was the presence of missing data, which required us to drop a significant number of data points before fitting the model. This resulted in a loss of information that could have potentially improved the accuracy of the model. In the future, we would like to explore alternative methods for dealing with missing data, such as imputation techniques, which would allow us to retain more of the data and potentially improve the performance of our model.

Another limitation we encountered during the development of our model was the large number of variables with unclear names. This made it difficult to understand the meaning and relevance of each predictor, which hindered our ability to identify the most important variables for the model. Additionally, the large dataset and our limited knowlege of the specifics of climate change and weather forecasting, coupled with the time constraints we were working under, made it challenging to fully understand the significance of each variable and how they would affect the performance of the model. This could have potentially affected the accuracy of the model. In future projects, we would like to have a better understanding of the data and variables, as well as more time to explore the data in depth to improve the understanding of the data and the model's performance.

We also ran into issues when hypertuning our parameters due to the large size of the dataset. It was difficult to run extensive hyperparameter tuning algorithms as the large dataset required a significant amount of computational power and memory. This made it difficult to run multiple tuning algorithms due to their extemely long runtime and the limited amount of available RAM. This limited our ability to find the optimal set of hyperparameters for the model and could have potentially affected the accuracy of our model. If we had more time, we would like to obtain access to more powerful computational resources to allow us to run more extensive hyperparameter tuning and improve the performance of the model. We also would like to gain more knowlege on how to optimize hyperparameters with such a large dataset.

Kaggle Submission

- We are currently 128 on the leaderboard
- Our best MSE is 1.525
- This is with our tuned XGBoost

Conclusion

In conclusion, our experience developing an XGBoost model was a valuable learning opportunity that resulted in a highly accurate model. In addition, it provided valuable insights and skills we will carry throughout our data science careers. XGBoost is a very useful tool for predicting long-term outcomes, and it can handle high-dimensional data, making it perform well for these types of competitions. Coming into the competition, our group only had a little prior experience with XGBoost, so we had to adapt to and overcome various challenges that arose throughout the process. For example, because the dataset was extensive and contained over 200 predictor variables, we had to learn new methods of hyperparameter tuning that would support this high dimensionality. Additionally, we gained experience cleaning and working with a disorganized, large dataset. Finally, we gained crucial experience collaborating with a team from different backgrounds and experience levels to achieve a common goal.

The model we developed with XGBoost has the potential to make a real-world impact by improving the field of weather forecasts and disaster preparedness. Using XGBoost, our model can produce highly accurate predictions of long-term outcomes, which can be especially useful in forecasting weather patterns. The current forecasting methods, primarily based on physics-based models, can be improved by incorporating insights from our XGBoost model. Predicting natural disasters well in advance is critical in addressing the challenges posed by climate change. Being able to anticipate severe weather events, such as hurricanes, floods, or droughts, allows us to take proactive measures to minimize the damage and loss of life.

Additionally, we can better allocate resources and plan for the future by predicting the likelihood of natural disasters. This is especially important as the effects of climate change continue to worsen. Implementing a machine learning model such as ours is an essential step in ensuring the safety and well-being of communities, as well as in addressing the more immense global challenges presented by climate change.