Weather Prediction Project

Objective:

This project is designed to test different prediction algorithms to determine the ideal model for predicting temperature in the short-term. It will use two main kinds of algorithms. The first are traditional statistics models. These are things like linear regression that will take the data and perform some computations with it. The second are machine learning models, which are more advanced models that are supposed to 'learn' based on the data.

What work has already been done:

What have others said the best model is?

Before making my own model, I wanted to see what other people have done and what their results were. Two studies helped me select the models as well as give some insight on what they found their best models to be.
Iliyas Ibrahim Iliyas' *Performance Evaluation of Machine Learning Models for Weather Forecasting* tested linear and ridge regression, as well as Random Forest regression. It claimed that Random Forest outperformed linear and ridge regression.
Md Saydur Rahman's *Comparative Evaluation of Weather Forecasting using Machine Learning Models* tested gradient boosting and random forest, with gradient boosting having a slight edge. I will be including both of these in my analysis.

How I hope to improve on the work others have done

My biggest problem with the sources I find is that they only judge models based on one thing; accuracy. This makes sense intuitively, but I think accuracy shouldn't be the only metric for judging. I will also add models I think might perform well, to see if there are any that were missed.

The models I will be using

I selected 8 models to be tested. They are the following
Statistical models:
Linear Regression, Ridge Regression, Lasso Regression, and Elastic Net.
Machine learning models:
Random Forest, Catboost, XGboost, and Gradient boost.
I chose CatBoost due to it being very popular for time sensitive predictions, and I haven't seen

anyone use it for weather forecasting. I also chose XGBoost due to it being intended to be an optimized version of Gradient boosting, so I wanted to see if it really could outperform Gradient Boosting. Gradient boosting and random forest were chosen due to them being popular for weather forecasting.

Procedure:

Data Processing

The weather data used in my learning was downloaded off of Brookhaven National Lab's meteorology website. I didn't know what factors influence temperature the most, so I came into contact with my school's meteorologist and had a conversation with him about what factors are the most important when it comes to weather forecasting. I was told that wind speed and direction, solar coverage, and temperature were the most important, so I downloaded those. My training data totalled to 10 years worth of weather data. With the data, I could start preprocessing. This was done entirely in python, using pandas to create the csv into a database. I preprocessed the data so my machine learning models could learn off them. The most important part of preprocessing is making lagged features. This gives the model a better idea of what happens to temperature as time continues, and this is the basis where it makes its predictions from.

Training the models

The models were then all loaded into a method designed to set it up for training. Each were trained on the same data then tested. This will give a good idea of how well trained our model is.

Grading Criteria

While accuracy is very very important for selecting a model, I thought it shouldn't be the only thing to consider when selecting a model. Due to this, I have selected 4 criteria to judge my models based on, each given a score between 0 and 5 except accuracy, which will count twice due to it still being the most important.

1. How accurate is the model?
2. How easy is the model to use?
3. How fast does this model train and predict?
4. How much memory does this model take up?

With this, we can start judging the models.

Results:

For all of the criteria, the lower the number the better.

Statistical models:

Linear Regression:
Training and prediction time: 1.72 seconds
Mean Absolute Error (MAE) on test data: 1.22
Memory used for training: 107.24 MB

Lasso Regression:
Training and prediction time: 12.57 seconds Final
Mean Absolute Error (MAE) on test data: 1.37
Memory used for training: 102.86 MB

Ridge Regression:
Training and prediction time: 0.64 seconds Final
Mean Absolute Error (MAE) on test data: 1.22
Memory used for training: 106.05 MB

Elastic Net:
Training and prediction time: 18.47 seconds Final
Mean Absolute Error (MAE) on test data: 1.34
Memory used for training: 101.87 MB

Ensemble Models:

CatBoost:
Mean Absolute Error (MAE): 0.9818742944603461
Memory used for training: 37.06 MB
Training and prediction time: 5.74 seconds

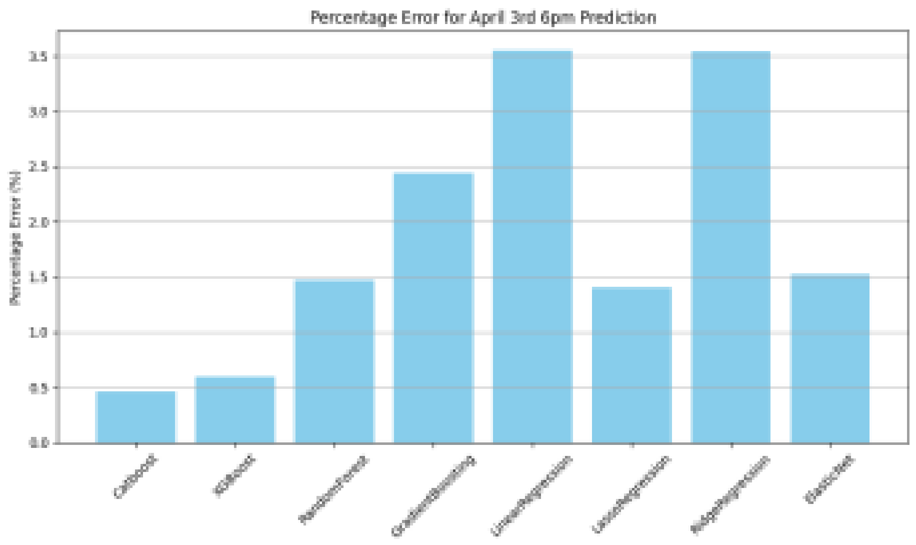
XGBoost:
Mean Absolute Error (MAE): 0.9884412312867559
Memory used for training: 2.11 MB
Training and prediction time: 1.64 seconds

Gradient Boosting:
Mean Absolute Error (MAE): 1.0421139119728235
Memory used for training: 4.09 MB
Training and prediction time: 89.50 seconds

Random Forest:
Mean Absolute Error (MAE): 0.957478703276419
Memory used for training: 1702.06 MB
Training and prediction time: 721.34 seconds

Models	Linear	Lasso	Ridge Elastic	CatBoost	Gradient	XGBoost	Forest
Speed	5	2	5 2	4	0	5	0
Memory	1	1	1 1	4	4	5	0
Accuracy	6	4	6 4	8	8	8	10
Simplicity	5	5	5 5	2	3	2	3
Total	11	12	17 12	18	15	20	13

Percentage error for a recent prediction:



Conclusion:

In general, it seems like machine learning models *are* better than statistical models. They had the best accuracy, but surprisingly also took the least amount of memory while oftentimes being faster to train as well. This was very shocking to me since I expected simpler mathematical operations to take much less memory and would be faster to execute. This might be due to the more advanced models having memory and time saving optimizations under the hood. XGBoost was the best model overall, followed closely by CatBoost. These two models were very fast, took little memory, and gave great predictions. Random Forest was by far the worst out of the ensemble models. Not because of the accuracy, but because of how tedious it was to train. It was actually the most accurate model I tested, but it took over a gigabyte of memory to train, as well as being saved in a file over a gigabyte large, which is gigantic compared to the megabytes the other models needed. Not only that, it took by far the longest to train, taking over 12 minutes compared to the few seconds required from the other models. These are factors I think are very important, which is why I wanted to test multiple criteria.

References:

Model inspiration from Maarten Laureyssen’s weather model project:
https://medium.com/@maarten./developing-a-weather-model-with-machine-learning-in-python-ed1b741dc553

More model inspiration as well as help predicting results from: Iliyas Ibrahim Iliyas’s research on the topic:
https://www.researchgate.net/publication/366163369_Performance_Evaluation_of_Machine_Learning_Models_for_Weather_Forecasting/link/6393d353095a6a777419f97a/download?_tp=eyJjb250ZXh0Ijp7ImZpcnN0UGFnZSI6InB1YmxpY2F0aW9uIiwicGFnZSI6InB1YmxpY2F0aW9uIn19

Analysis help from Md Saydur Rahman’s Comparative Evaluation of Weather Forecasting using Machine Learning Models:
https://arxiv.org/abs/2402.01206