**Emerging Technology**

**Final Project**

**Report**

**CSTP 2301 – Emerging Technologies**

**Group Members:**

**Betty Merid**

**Jean Andrei Amante**

**Machine Learning Models Used:**

1. Ridge Regression
2. Multilayer Perceptron (MLP)
3. Gradient Boosting
4. Random Forest
5. XGBoost

The goal is to use different models and the given dataset to forecast the future values after the last recorded month. We realized that using these model/algorithms we would be able to get a decent accuracy for each selected feature. First, Ridge Regression, this algorithm can handle multicollinearity which means it can handle when features in the data are highly correlated. In this code, Ridge Regression is used rather than Linear Regression because it adds a regularization term, which helps prevent overfitting and improves generalization to the unseen data. Secondly, Multi-layer Perceptron, this algorithm can catch complex patterns which makes it suitable for nonlinear relationships. Next, Gradient Boosting and Random Forest. These two models are ensemble learning techniques that use multiple models to help increase the prediction accuracy. Gradient Boosting prioritizes decreasing the mistakes or errors iteratively, and Random Forest prioritizes voting or averaging. Lastly, XGBoost, provides better performances and efficiency. We believe that the models we decided to use are good for time series data because they can handle nonlinear relationships and complex patterns.

**Summarized Result**

**Feature: 542236**

|  |  |
| --- | --- |
| **Name** | **Accuracy** |
| Ridge Regression | 100.00% |
| MLP | 98.85% |
| Gradient Boosting | 91.61% |
| Random Forest | 90.91% |
| XGBoost | 90.98% |

**A graph showing different colored lines

Description automatically generated**

**Feature 67321**

|  |  |
| --- | --- |
| **Name** | **Accuracy** |
| Ridge Regression | 99.95% |
| MLP | 74.74% |
| Gradient Boosting | 82.87% |
| Random Forest | 82.50% |
| XGBoost | 82.84% |

A graph showing the results of a performance

Description automatically generated with medium confidence

**Feature 549295**

|  |  |
| --- | --- |
| **Name** | **Accuracy** |
| **Ridge Regression** | **98.43%** |
| **MLP** | **33.52%** |
| **Gradient Boosting** | **94.20%** |
| **Random Forest** | **92.91%** |
| **XGBoost** | **93.67%** |

**A graph with different colored lines

Description automatically generated**

**Feature 41108**

|  |  |
| --- | --- |
| **Name** | **Accuracy** |
| **Ridge Regression** | **99.99%** |
| **MLP** | **74.72%** |
| **Gradient Boosting** | **92.23%** |
| **Random Forest** | **91.79%** |
| **XGBoost** | **91.75%** |

A graph with colored lines

Description automatically generated

**Feature 541982**

|  |  |
| --- | --- |
| **Name** | **Accuracy** |
| **Ridge Regression** | **100.00%** |
| **MLP** | **93.29%** |
| **Gradient Boosting** | **94.96%** |
| **Rnadom Forest** | **94.45%** |
| **XGBoost** | **93.55%** |

**A graph of a graph

Description automatically generated with medium confidence**

**\*\*The best model is Ridge Regression with an accuracy of 100.00%\*\***

**What Helped Improve the Models/Accuracy?**

1. Hyperparameters

* Using hyperparameters is important because they are the settings that we can choose before training a machine learning model; thus, why we chose to add hyperparameters in our models. In our code, parameters such as the regularization strength in Ridge Regression, the number of hidden layers and neurons in MLP, the number of trees and learning rate in Gradient Boosting and Random Forest, and parameters in XGBOOST helps the models to get different data patterns and relationships. By changing/tuning these parameters, we can control the model’s complexity, stop overfitting, and optimize the model’s performance.

1. Regularization

* Adding regularization in our code helps us to prevent overfitting. This is also the reason why we chose to use Ridge Regression over Linear Regression because Ridge Regression gives an extra element to the loss function, reducing the model form learning overly complicated relationships. Moreover, regularization helps to clean the model’s predictions and reduce impact of noisy data points which leads to better generalization on test data. The predictions are also more reliable in the context of time series analysis.

1. Normalization

* The goal of normalization is to transform features to be on a similar scale which will then improve the performance and training stability of the model. Feature scaling, such as the normalization with StandardScaler, helps improve machine learning models by making sure that all input features are on a similar scale. It also makes sures that there is no single feature that dominates the others in influencing the model’s prediction. It reduces redundancy and improve integrity.

**Code Explanation**

This code iterates over each feature in the “features” list. The “X” is the input data, and “y” is the target data for the current feature. We reshape the ‘X’ to have a single feature

A black background with blue and white text

Description automatically generated

The data is split into training set and testing set, and the last 36 data points are kept for testing.

A black background with numbers and symbols

Description automatically generated

This is a common preprocessing step that ensures that all features have the same scale. “StandardScaler” standardizes the features by removing the mean and scaling to unit variance. The second line scales the training features by fitting the StandardScaler class to them and transforming the data. It also computes the mean and standard deviation of the features in the training set and then applies the transformation to standardize the data. The third line scales the test features using the mean and standard deviation obtained from the training set. This ensures that both training and test sets are scaled consistently.

A computer code with text on it

Description automatically generated

This code help improve the model performance and reduce computational complexity. PCA is used and applied to reduce the dimensionality of the data.

A computer code on a black background

Description automatically generated

This loop iterates over each model in the “models” dictionary which contains all the five models that we are going to use. Each model is trained on the training data, X\_train and y\_train, and then they are used to make predictions on the test data, X\_test.

A screen shot of a computer code

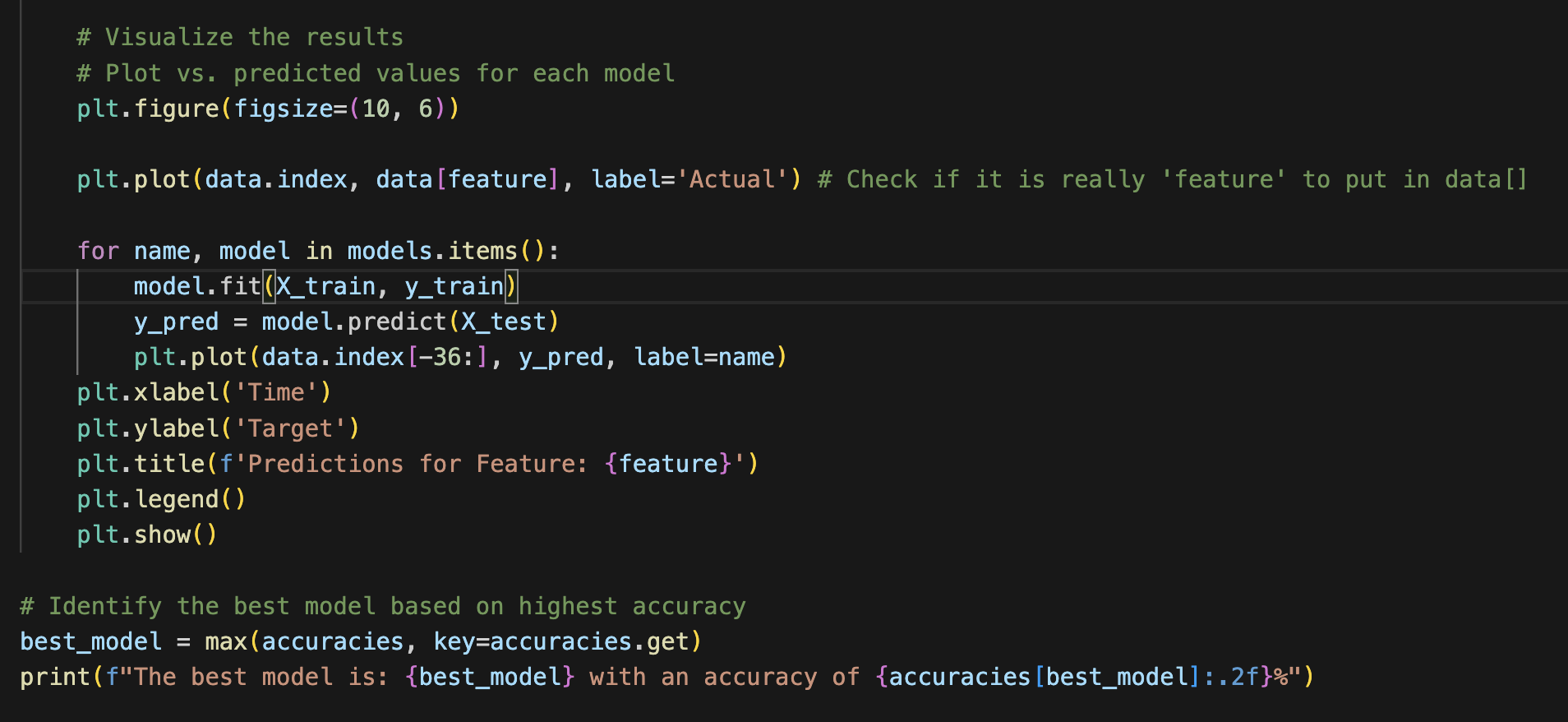
Description automatically generated

This code will print out the accuracy of each model for the current feature.

A black background with colorful text

Description automatically generated

This code plots the actual values of the feature over time and blends the predictions made by each model. At the very bottom, after evaluating all the features, the code determines the best model based on the highest accuracy achieved. It will print out the name of the best model with the accuracy.



**Improvement Suggestion:**

We looked more into the hyperparameters that we used after finishing the project. We had some concerns about overfitting in our model, so we spent some time seeing how changing the hyperparameters changed our accuracies to make it better.

**Week 1-2 report**

So far we have been working on the data preparation, we had some trouble understanding what exactly was expected so we had done our codes a little wrong. So for now we are working on just a single column and using the different regression models on just that to make sure we are understanding and doing the codes right. After we are able to do this correctly we will work on adding the other 4 columns into the code. The regression models we wanted to try for now are

* Liner regression
* Gradient boosting
* MPL
* SVR

So far from our understanding and by talking with classmates for using the 5 different columns to test the models we need to use a loop. We would make an array of sorts with the different column inside, then we would loop through them and use the different regression models on them.

**New Submission (MLP Regression)**

**Observation:**

The accuracy changes when the number of neurons, number of hidden layers, max iteration, epochs, and activation function changes. Basically, when the hyperparameters changed, the accuracy changes as well. We tried and changed these hyperparameter to get higher accuracy and we realized that there are at least thousands or hundreds of possible patterns of these hyperparameters.

Multicollinearity:

<https://statisticsbyjim.com/regression/multicollinearity-in-regression-analysis/>

Ridge and Lasso Regression:

<https://www.analyticsvidhya.com/blog/2017/06/a-comprehensive-guide-for-linear-ridge-and-lasso-regression/>

MLP:

[https://www.datacamp.com/tutorial/multilayer-perceptrons-in-machine-learning#](https://www.datacamp.com/tutorial/multilayer-perceptrons-in-machine-learning)

Gradient Boosting:

<https://www.numpyninja.com/post/gradient-boost-for-regression-explained>

Normalization:

https://www.linkedin.com/advice/0/how-can-normalization-improve-database-performance-yv9cf#:~:text=Normalization%20is%20a%20method%20of,referential%20integrity%2C%20and%20simplifying%20queries.