Unemployment Rate Forecasting using Machine Learning

Susan Jiao
jjiao25@wisc.edu

Yuanhang Wang
wang2243@wisc.edu

Yi Xiao
yxiao84@wisc.edu

Abstract

Building accurate forecasting models for economic indicators is a research area that many policy researchers work on. Traditional time series forecasting methods such as autoregressive moving average (ARMA) models often lead to unsatisfactory results. In this project, we exploit machine learning and deep learning techniques to forecast unemployment rate. We use three months of data to predict the following one month. Using linear regression as a baseline model, we compare results from random forests, XGBoost, and long short-term memory. There are two variants in all models: one uses 11 relevant economic indicators as input features, while another uses unemployment rate as the only feature. Both mean squared error (MSE) and mean absolute error (MAE) are used as evaluation metrics. We exclude year 2020 to control for noise from the COVID-19 pandemic. Among models that utilize all 11 features, XGBoost gives the best performance with MSE of 0.055. Among models that use unemployment rate as the only feature, baseline linear regression performs the best. In addition, we test our models on year 2020 to see their performance during unusual time, we find XGBoost and LSTM to perform the best in the variant that uses all 11 features.

1. Introduction

Policy researchers conduct extensive economic forecasting to make good suggestions to policy makers. However, forecasting economic indicators is challenging, since indicators can be influenced by a wide-range of factors such as politics and financial markets. The task has been especially difficult in today's highly interconnected and dynamic economy. For instance, predicting the next recession has been a research question that many economists and think tanks work on. Researchers typically utilize a combination of relevant economics indicators based on economic theory and apply statistical models to make predictions. A common predictor for recession would be the negative spread between the yield on 10-year treasury security and the yield on 1-year treasury security, and the most common models used are linear and nonlinear regression models. However, the predicted results often turn out to be unsatisfactory. One explanation could be that the underlying economic hypothesis needs to be refined, while another explanation could be that there are rooms for improvement on the statistical models. In this project, we focus on building various statistical models and identify the model that best predicts the results.

Unemployment rate is one of the most common and important economic indicators. It is a direct indicator of a country's labor market performance and reflects the general health of the economy. More importantly, it is a significant factor in many government decision making processes, such as setting interest rates and designing social welfare programs. One closely relevant social welfare program in the current pandemic period is unemployment insurance, a federally mandated, state-run program in which payroll taxes are used to pay benefits to unemployed workers laid off by companies. In general, eligible individuals may collect unemployment insurance for 6 months. In recessions, benefits are automatically extended to 9 month or 12 months and may be further extended in deep recessions. However, setting a benefit period that is too long might result in moral hazard problem. That is, unemployed individuals might have less incentive to search for a new job. Accurate estimation of the change in unemployment rate could help policy makers adjust the benefit period that best balances the trade-off between risk protection and moral hazard.

Building accurate statistical models that forecast unemployment rate and economic indicators in general could help policymakers make better policy decisions that stabilize the economy. In this project, we forecast unemployment rate by utilizing machine learning techniques such as random forests and neural networks, and we use standard regression methods as benchmark models.

2. Related Work

The most standard methods of time-series forecasting are moving average (MA) models and autoregressive (AR) models which predict future observations using recent observations[3]. Some commonly used ones are threshold autoregressive (TAR) models, Markov switching retrogressive (MSA) models, linear univariate autoregressive integrated moving average (ARIMA) model, and bivariate

vector autoregressive moving average (VARMA) models. Those models are widely used in macroeconomic indicator forecasts. Besides statistical models, consensus forecasting is another methodology being used. Instead of using one specific model, it combines different models that draws from a variety of techniques[5].

Those traditionally used methods for unconditional forecasting are mostly linear models that have high data demands. More recently, researchers in economics and political science begin to utilize deep learning techniques, particularly neural networks such as Long Short Term Memory (LSTM) and Gated Recurrent Unit (GRU), to make predictions. Researchers in economics find deep neural networks to achieve higher accuracy with limited data than traditional econometrics methods do. Cook and Hall (2017) predict civilian unemployment using models that built upon four different neural network architectures, and those models outperform consensus forecasting at short time horizons. In particular, they find that an Encoder Decoder architecture outperform baseline models at every forecast horizon.

3. Proposed Method

3.1. Problem Description

This project aims to predict the unemployment rate using economic indicators of the preceding months. It is a multi-variate time series forecasting problem. Specifically, we explore the task of using data from the previous quarter (3 months) to predict the unemployment rate of a month. In this project, we fit two different kinds of machine learning models to forecast unemployment rate.

We use the following method to solve this problem: firstly, we use a linear regression with all features as our baseline model. After that, we conduct feature selection and try different machine learning methods, including Linear Regression, Random Forest, XGBoost, and a deep learning model LSTM. These proposed models are trained on the same training sets and can compare to each other. We also take a look at the Autoregressive Integrated Moving Average (ARIMA) model. However, ARIMA deals with a different task and cannot be compared to other models.

3.2. Data Preprocess

Time series data has a special sequential structure. In our task, we assume that the unemployment rate of each month can be fully expressed by the economic indicators of the previous 3 months. However, each month has different values for each indicator, giving an $n * 3$ matrix as features, where n is the number of economic indicators including unemployment rate. Therefore, we need to process our data to construct training/test sets that can be used for supervised learning models.

Suppose we have data for m months $D = \{d^{(t)}\}_{t=1}^{m}$ with n economic indicators, i.e. $d^{(t)} = [d_1^{(t)}, ..., d_n^{(t)}]^T$. We construct $n - 3$ data points using a sliding window of length 4. Starting from the beginning, the first data sample contains the $3 * n$ values of the economic indicators from the first 3 months as features and the Unemployment rate of the 4-th month as label. Each time, we slide the window forward by 1 month and obtain a new data sample. I.e., $x^{(i)} = [d_1^{(i)}, ..., d_n^{(i)}, d_1^{(i+1)}, ..., d_n^{(i+1)}, d_1^{(i+2)}, ..., d_n^{(i+2)}]$, a $3n * 1$ vector.$y^{(i)} = d_{Unemp}^{(i+3)}$. Hence, we construct our data set as $\mathcal{D} = \{(x^{(i)}, y^{(i)})\}_{i=1}^{m-3}$. The design matrix $X \in \mathbb{R}^{(m-3)*(3n)}$.

3.3. Feature Selection

We select features in two ways. Initially we select features depending on the correlation of each feature with our target unemployment rate. We check the absolute value of the correlation and preserved all features with absolute correlation greater than 0.5. This way, we select 10 economic indicators as features: CUMFNS, BAAFFM, CLAIMSx, HWIURATIO, UEMPLT5, UEMP5TO14, UEMP15OV, UEMP15T26, UEMP27OV, CES1021000001. The description of these indicators can be find in Appendix A of this report. Combined with UNRATE (Unemplotyment rate), we have in total 11 features as our initial selection.

We also fit a random forest regressor on the 11 features selected and look at the feature importance. However, more than 95 percent of weights are given to UNRATE from the previous month. This result is not of much interest, so we decide to use the 11 initially selected features.

3.4. Linear Regression

Linear regression is the simplest machine learning model. It fits a straight line to the data. Linear regression is computationally efficient (it has a closed form solution) and easy to interpret, so it serves as our baseline model.

3.5. Random Forest

We select the random forest regressor as the first model to compare with the baseline linear regression model.

To implement the random forest algorithm appropriately for time series forecasting, we first transform the time series dataset we obtained into a supervised learning problem using a sliding-window representation.

We perform holdout validation to determine the best value of the hyperparameter: $bootstrap$, max_depth, $max\_features$, $min\_samples\_leaf$, $min\_samples\_split$, and $n\_estimators$ in the random forest algorithm. We do not use k-fold cross-validation because it would not be valid to fit a model on data from the future to predict the past. Since we do not use cross-validation, we input a predefined split for the cross-validation parameter. The training dataset is split into five sub-partition. As a result, the model is

trained using 4 of the sub-partition and validated using the remaining one partition of the training set. We use gridsearchCV to find all the best hyperparameters in one time, and the best value of the parameters for random forest listed above are determined.

In addition, bagging is applied to the training process of the model. Bagging is a learning algorithm that fits model using different subsets of the dataset, subsets are randomly selected by using bootstrap. The variance of the model would be reduced through bagging.

3.6. XGBoost

Apart from using a simple random forest algorithm, we choose to fit an XGBoost model to compare with the baseline Linear Regression model. This model also transforms the series datasets into supervised learning using a sliding-window representation for further forecasting.

XGBoost is an ensemble of decision trees algorithm where new trees fix errors of those trees that are already part of the model. Trees are added until no further improvements could be made to the model. This model provides an efficient implementation of the gradient boosting algorithm and access to a set of model hyperparameters designed to control the model training process.

We also perform holdout validation to determine the hyperparameters in the XGBoost model. In the holdout validation, we use a multi-step hyperparameter tuning to avoid the excessive calculation and complexity caused by too many parameters. Since we do not use cross-validation, we also input a predefined split for the cross-validation parameter in this model.

3.7. LSTM

Recurrent Neural Network (RNN) is the most common deep learning model used for time series analysis. The unemployment rate data fits well in this context. However, RNN is difficult to train due to the exploding and vanishing gradient problem. LSTM is an improved architecture built from RNN. It deals with the vanishing gradient problem by reparameterizing the RNN. LSTM consists of three gates: input, out, and forget. The gates facilitate the cell to remember values for an arbitrary amount of time and control the flow of information in and out of the LSTM cell. LSTM has a rather complex structure, but it tends to have good prediction performance. An example would be application in financial time series prediction problems. Fisher and Krauss (2018) use LSTM to predict directional movements for the constituent stocks of the S&P 500. They find LSTM networks to outperform memory-free classification methods, including random forest, deep neural net, and logistic regression classifier [2].

Consistent with random forest and XGBoost, we first transform the unemployment dataset into supervised learning using a sliding window representation. We then fit the data into a LSTM model to see whether it outperforms previous models.

3.8. ARIMA

Autoregressive Integrated Moving Average (ARIMA) is a traditional statistical method to model time series. We use univariate Seasonal ARIMA to model the unemployment rate, not considering other economic indicators. We simply fit an ARIMA model on the training set and generated predictions on the test set, as it does not make sense to fit an ARIMA model on a dataset of 3 month and forecast the unemployment rate of the next month. A dataset with 3 data points is too small for an ARIMA model. ARIMA model also has other limitations. It assumes the data to have constant variance and models the mean of the data. As the unemployment rate barely has constant magnitude of variation. Another reason that we don't compare ARIMA to machine learning models is that they have different objectives. In this regression analysis problem, our machine learning methods aims to minimize the forecast MSE of the predicted unemployment of the label month, but ARIMA simply finds the best fit to the time series by minimize AIC or log likelihood all the data in the training set. They have intrinsically different goals.

4. Experiments

4.1. Dataset

We obtain our data from Federal Reserve Economic Data [4]. The dataset contains 129 different economic indicators recorded monthly from Jan 1959 through Aug 2020, including the unemployment rate.

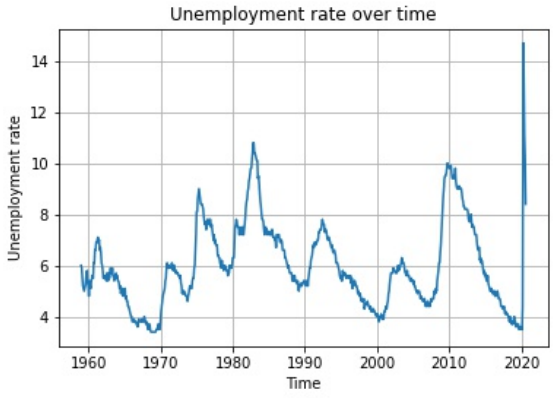


Figure 1. Unemployment rate

We observe from Figure 1 that the data for 2020 is unexpectedly high due to the pandemic, so we conduct two

sets of experiments. The first experiment includes data from 2020, and the second experiment excludes data from 2020.

4.2. Random Forest

We build two random forest models, one model only has UNRATE as feature, and another model has the 11 features we selected from the feature selection.

In the model with one feature. We split the transformed dataset into training set(0.6), validation set(0.2), and test set(0.2). To choose the best parameters, we use the RandomizedSearchCV to narrow the range for the best parameters. We create a random grid by randomly selecting a range (200, 400, 600, ..., 2000) for the parameter $n\_estimators$, and setting two choices $['auto','sqrt']$ for the parameter $max\_features$, selecting a range (10, 20, ... 110, None) for the parameter max_depth, and choosing (2, 5, 10) for min_sample_split. After narrowing the range for best parameter values, we create a parameter grid based on random search results. However, we get a worse result after using GridSearchCV. So, we use the best parameters from the random search since the result is ideal. After all parameters are selected, we test the model on the test dataset, obtain the final test MSE and test MAE, and obtain the train MSE and train MAE to identify overfitting or underfitting.

In the model with 11 features. We split the transformed dataset with the same proportion above. We also use the RandomizedSearchCV to get a narrower range for the best parameters to choose the best parameters. After narrowing the range for best parameter values, we create a parameter grid based on a random search. We use the best parameters from the Grid Search Cross-validation. After all parameters are selected, we test the test dataset model, thereby obtaining the test dataset and obtaining the final test score.

4.3. XGBoost

We also build two XGBoost models, one has UNRATE as the only feature, and another one has the 11 features as we selected before.

In the model with only one feature, we input a predefined split for cross-validation parameter since we do not use cross-validation. In the first step, we find the best value for the parameter max_depth is 7, and for min_child_weight is 3. After the first tune, we get the train MSE of 0.025 and the validation MSE of 0.0200. In the second step, we find the best value for the parameter $gamma$ is 0.4. The validation MSE has decreased to 0.0199. In the third step, we get the best value for the parameter $subsample$ is 0.7 and for the $colsample\_bytree$ is 0.7. The MSE for the Validation set has slightly decreased. In the last step, we find the best value for the reg_alpha is 0.0. The validation MSE also has slightly improved.

In the model with 11 features, we also input a predefined split for the cross-validation parameter. In the first step, we aim to find the best max depth and min child weight. We set a possible range (3, 5, 7, 9) for the best value of parameter max_depth, and set a possible range (1, 3, 5) for the best value of parameter min_child_weight. After the first tune, we get the train MSE of 0.0052 and the validation MSE of 0.0340, as the max_depth set to 9 and the min_child_weight set to 1. Then we tune the parameter max_depth separately, the best value for the parameter is still 9 but the Validation MSE decreases to 0.0293. In the second step, we tune the parameter $gamma$ with a setting range (0, 0.1, ..., 0.5)for $gamma$. We get the best value for $gamma$ is 0.0. The Validation MSE does not change much after this tunning. In the third step, we tune the parameter $subsample$ and $colsample\_bytrees$ a setting range (0.6, 0.7, ..., 1) to $subsample$ and (0.6, 0.7, ..., 1) to $colsample\_bytrees$. We get the best values for $colsample\_bytree$ is 0.8 and for $subsample$ is 0.9. The validation MSE has decreased to 0.0269. In the last step, we tune the parameter reg_alpha, and we get the best value for reg_alpha is 0.01. The validation MSE has decreased to 0.0267.

4.4. LSTM

Using one feature to make prediction is not suitable in LSTM model. We use 11 features in the LSTM model. Keras is used to implement the LSTM model.

We use LSTM with 32 units as the input layer. The activation function we use is Relu. We have compared results among tanh, sigmoid, and Relu, and we find Relu to perform the best. In addition, we add a dropout with probability 0.6 after each layer to reduce overfitting. A total of two LSTM layer is implemented.

However, despite setting high dropout probability after each LSTM layer, the model still suffers from serious overfitting problem. One reason could be that our dataset is monthly data which is much smaller than datasets in other applications of neural networks. Besides dropout, two other regularization techniques can be used to address overfitting in neural networks: early stopping and L1/L2 regularization. We will implement those two techniques in future work on the project.

Different from the previous three models in which we use three month of data to make a single step prediction on the next month, we also use one month of data to predict the next month in the LSTM model. We find the overfitting problem reduced to a great extent. We discuss the LSTM result in this variant of the moving window size separately in the result section. In one-month window size, we implement a single layer LSTM with 32 units with Relu as the activation function. Overfitting is small, and adding dropout layer does not further reduce overfitting.

4.5. Software

For model training, we mainly use Jupyter Notebook, Google Colab, and DeepNote.

4.6. Hardware

Each group member uses his/her own laptop.

5. Results and Discussion

We mainly use the mean squared error(MSE) and the mean absolute error(MAE) to evaluate our models. They are common metrics for regression analysis.

$$MSE = \frac{1}{N}\sum_{t=1}^{N}(y_t - \hat{y}_t)^2 \quad (1)$$

$$MAE = \frac{1}{N}\sum_{t=1}^{N}|y_t - \hat{y}_t| \quad (2)$$

Our data set has 129 columns, each representing an economic indicator. For any missing data, we simply drop the row or column without imputing. This is because most of the missing values are consecutive, e.g. missing all values from Jan 1959 to Jan 1960. It is hard to interpolate consecutive missing values. Also, it does not make much sense to impute the missing values with the mean or the median. This is because 1) the economic indicators have huge fluctuations, and 2) we only use data from 3 months as features, so the imputed values will carry large weights and any error due to the imputation will significantly affect the result. Our selected 11 features only contain a few missing Another merit of this dataset is that all the economic indicators are numerical, so we do not have to deal with categorical variables.

5.1. Linear Regression

For the Linear Regression (LR), we simply drop all the missing values and use all 129 features. We window the data and directly feed it to the a linear regression model. The performance was very bad, as there are too many features, resulting in overfitting. The MSE is 0.0075 on training set and 162.547 on test set. Note that here we are not actually predicting the the unemployment "rate", which is a number between 0 and 1, but the unemployment rate * 100.

Besides the baseline model, we also fit LR models only using the Unemployment Rate and using the 11 selected features. The performance of Linear Regression is unexpectedly high in this task. For the dataset excluding data from 2020, LR with 10 features has an MSE of 0.094 and LR with only 1 feature has an MSE of 0.028. We see that the LR model is still overfitting the data as the 11-features performance is lower than using only 1 feature, though.

Figure 2. Random Forest Model (UNRATE as the only feature) Test Result

5.2. Random Forest

For the random forest model with UNRATE as the only feature, we find that it gives us the best result when the number of trees in random forest ($n\_estimators$) is 200, the maximum number of levels in tree (max_depth) is 10, the minimum number of samples required to split a node($min\_samples\_split$) is5, the minimum number of samples required at each leaf node ($min\_samples\_leaf$) is 4, the method of selecting samples for training each tree is bootstrap. After fitting this model and using this model predict UNRATE, we obtain a train MSE of 0.0217, a train MAE of 0.1095, and the test results are showing in the Figure 2.

For the random forest model with 11 features, we find that it gives us the best result when the number of trees in random forest ($n\_estimators$) is 1200, the maximum number of levels in tree (max_depth) is 100, the minimum number of samples required to split a node($min\_samples\_split$) is 2, the minimum number of samples required at each leaf node ($min\_samples\_leaf$) is 3, the method of selecting samples for training each tree is bootstrap. After fitting this model and using this model predict UNRATE, we obtain a train MSE of 0.0107, a train MAE of 0.0727, and the test results are showing in the Figure 3. The random forest model with 11 features is more overfitting than the random forest model with UNRATE as the only feature.

5.3. XGBoost

For the XGBoost model with UNRATE as the only feature, we implement hyper-parameter tuning in multi-step. In this first step, we tune the max depth and min child weight. In the second step, we tune gamma. In the last step, we tune the subsample and $colsample\_bytrees$. We find that it gives us the best result when the hyper-parameter max_depth is 7, min_child_weight is 3, $gamma$ is 0.4, $colsample\_bytree$ is 0.7, $subsample$ is 0.7, and reg_alpha

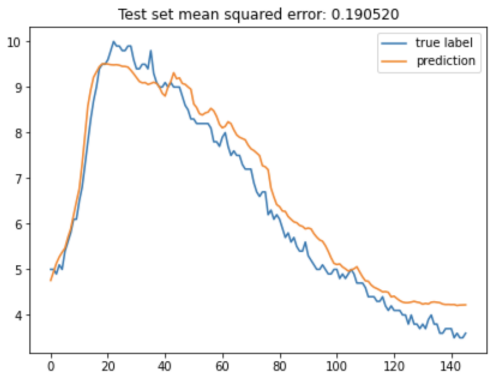


Figure 3. Random Forest Model (with 11 features) Test Result

Figure 5. XGBoost Model (with 11 features) Test Result

Figure 4. XGBoost Model (UNRATE as the only feature) Test Result

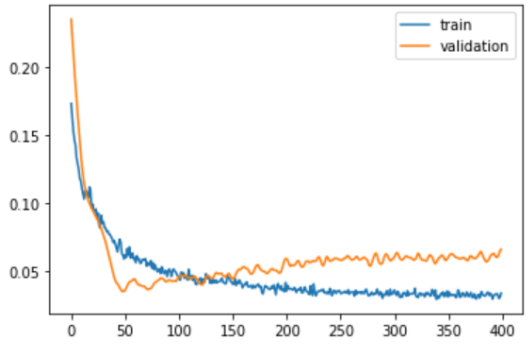


Figure 6. LSTM loss function using three months to predict one month (year 2020 excluded)

is 0.0. After fitting this model and using this model to make a prediction, we obtain a train MSE of 0.033, a train MAE of 0.136, and test results are showing in the Figure 4.

For the XGBoost model with 11 features, we also implement a multi-step hyper-parameter tuning. We find that it gives us the best result when the hyper-parameter max_depth is 9, min_child_weight is 1, $gamma$ is 0.0, $colsample\_bytree$ is 0.8, $subsample$ is 0.9, and reg_alpha is 0.0. After fitting this model and using this model to make predictions, we obtain a train MSE of 0.0159, a train MAE of 0.0953, and test results are showing in the Figure 5.

5.4. LSTM

Figure 6 shows the result that use 3 months of data to predict future one month. As discussed in section 4.4, this moving window size suffers from serious overfitting problem in LSTM. The results are therefore poor as expected. The fourth row of table 1 shows the result. We can see that it performed the worse among all models. This is unexpected, because we have discussed at the beginning of the report that LSTM has shown good forecasting performance in the time series forecasting literature. We could potentially improve this result by implementing L1/L2 regularization besides adding dropout.

Figure 7 shows the result that use 1 month of data to predict the next month. Overfitting problem is reduced to a great extent. We note here that dropout is not used, and adding dropout layer does not further reduce overfitting and indeed leads to lower performance. The fifth row of Table 1 and 2 display those results. For data that excluded year 2020, the MSE on test set is 0.067, and the MAE is 0.231. For data that included year 2020, the MSE is 0.687, and the MAE is 0.313. Though we again note that the window size here is different from other models, LSTM gives the second best performance among all models.

5.5. ARIMA

We will briefly discuss the results of the ARIMA model, as it is not comparable to other models. The ARIMA model deals with a different task: it fits all the data in the training set, and we generate forecast on all the test data.

We find no clear pattern of Seasonality in the series, so we fit a regular ARIMA model. Our best fitted ARIMA model is ARIMA(5,1,9), with an AIC of -434.431. The

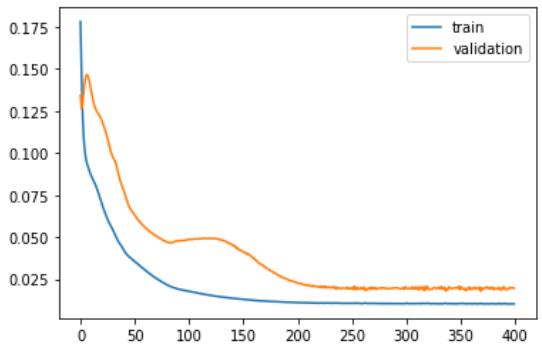
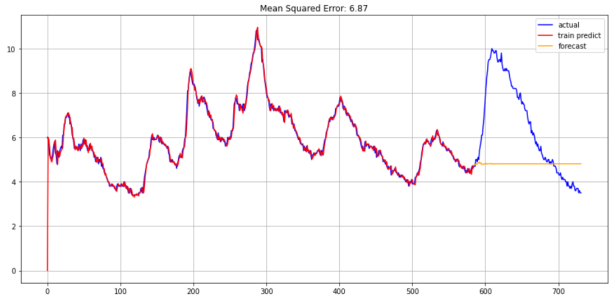


Figure 7. LSTM loss function using one month to predict one month (year 2020 excluded)

residuals are not correlated and have constant variance around zero. Therefore, we consider ARIMA(5,1,9) to be a good fit to the training data. The performance of ARIMA can be assessed in the following graph.



The problem of ARIMA model is evident on the graph. It only models the mean of a time series, so the forecast values are around the mean of the series with decreasing variance, and finally converges to the mean, giving bad performance. A better way to fit the unemployment rate would be first transform the data with some functions, such as log or $log\ difference$.

5.6. Summary of results

By now we have discussed all of the results. Table 3 and Table 2 summarizes our results. We note that LR has the best performance on average, though the performance of the models are actually very close. We think that the unexpectedly high performance of LR is due to the following reason: the unemployment rate of 4 months does not change much, so the relationship between the unemployment rate of one month and that of the previous 3 months is simple. Therefore, in this one-step forward forecast problem, LR achieved a high performance.

We note that if we change the task to a multiple-steps forward forecast, LR may not have this high performance. Essentially, as the task gets more complex, LR may not have the enough capacity to learn the complex relationships between the variables. This lack of capacity was not revealed in this simple, single-step forecast task. However, it might be a huge problem in multi-step forecast, as the task is more complex, and the prediction error accumulates in each step of forecast (when the predicted values in one step become the feature of the next step).

Another reason for the relatively bad performance of ensemble and deep learning models is the different distribution between the validation set and the test set. During the hyperparameter tuning phase, the MSE on validation set could be very low, but the error remains relatively higher on the test set. As we cannot use cross validation, it is hard to solve this shift in distribution of data. A potential way to alleviate the problem of shifted distribution would be creating the validation set with different size and average the performance, but it won't help much, as the validation sets used will contain similar data.

	One feature		Eleven features	
Method	MSE	MAE	MSE	MAE
Linear Regression	**0.028**	**0.130**	0.094	0.245
Random Forest	0.035	0.144	0.191	0.382
XGBoost	0.054	0.171	**0.055**	**0.190**
LSTM	-	-	0.489	0.610
LSTM (one month)[1]	-	-	0.067	0.231

[1] LSTM (one month) denotes using one month moving window size instead of three months to make prediction.

Table 1. Model performance using data from Jan 1959 to Dec 2019

	One feature		Eleven features	
Method	MSE	MAE	MSE	MAE
Linear Regression	0.938	**0.250**	**0.836**	0.365
Random Forest	**0.842**	0.261	-	-
XGBoost	0.892	0.280	0.933	**0.285**
LSTM	-	-	1.151	0.902
LSTM (one month)[1]	-	-	**0.496**	0.341

[1] LSTM (one month) denotes using one month moving window size instead of three months to make prediction.

Table 2. Model performance using data from Jan 1959 to Aug 2020

6. Conclusions

In our unemployment rate forecasting problem, we mainly explore the task of predicting the unemployment rate using economic indicators from the previous 3 months. We propose 4 methods to solve this task – Linear Regression, Random Forest, XGBoost, and LSTM – and compare their performance. We also had a quick look at the statistical model ARIMA.

Based on the result, LR has the best overall performance.

Besides, XGBoost also had a decent performance as expected. We believe that LR being the best model is because we are exploring a relatively simple task. There are two improvements that could be done in the future. Firstly, one can explore more complex tasks, such as multi-step forecasting (predicting the unemployment rate of n months ahead), expanding the time horizon (using 6 or 12 months as input), or using variable-length time windows. In addition, due to the lagged nature of economic indicators, the change in unemployment rate may not be full expressed by these indicators in the previous terms. We need to find more powerful indicators whose changes precede the change of unemployment rate.

7. Acknowledgements

We gratefully appreicate guidance and support from our advisor Dr. Sebastian Raschka. He has been a great mentor for our team.

8. Contributions

Susan found the original dataset from the web [1]that contains FRED-MD and FRED-QD databases for macroeconomic research. She also coded the deep learning neural network model (LSTM). Yuanhang Wang processed the original dataset and did feature selection. He also implemented Linear Regression models and the ARIMA model. Yi Xiao coded all other machine learning models, including Regression with Random Forest and XGBoost. All team members, Susan, Yuanhang, and Yi, worked together and finished the project report.

References

[1] Fred-md and fred-qd: Monthly and quarterly databases for macroeconomic research, 2015-2020.

[2] T. Fischer and C. Krauss. Deep learning with long short-term memory networks for financial market predictions. *European Journal of Operational Research*, 270(2):654–669, 2018.

[3] A. S. Hall. Machine learning approaches to macroeconomic forecasting. *The Federal Reserve Bank of Kansas City Economic Review*, 103(63):2, 2018.

[4] M. W. McCracken. "FRED-MD and FRED-QD: Monthly and Quarterly Databases for Macroeconomic Research". *Federal Reserve Bank of St. Louis*.

[5] A. Smalter Hall and T. R. Cook. Macroeconomic indicator forecasting with deep neural networks. *Federal Reserve Bank of Kansas City Working Paper*, (17-11), 2017.

9. Appendix

9.1. Appendix A

Due to the space limit, we are unable to give full descriptions of all 129 economic indicators. Here is the description of the 11 indicators we used.

fred	description	gsi	gsi:description
CUMFNS	Capacity Utilization: Manufacturing	M_116461602	Cap util
BAAFFM	Moody's Baa Corporate Bond Minus FEDFUNDS		Baa-FF spread
CLAIMSx	Initial Claims	M_15186204	UI claims
UNRATE	Civilian Unemployment Rate	M_110156541	U: all
HWIURATIO	Ratio of Help Wanted/No. Unemployed	M_110156531	Help wanted/unemp
UEMPLT5	Civilians Unemployed - Less Than 5 Weeks	M_110156527	U ¡ 5 wks
UEMP5TO14	Civilians Unemployed for 5-14 Weeks	M_110156523	U 5-14 wks
UEMP15OV	Civilians Unemployed - 15 Weeks and Over	M_110156524	U 15+ wks
UEMP15T26	Civilians Unemployed for 15-26 Weeks	M_110156525	U 15-26 wks
UEMP27OV	Civilians Unemployed for 27 Weeks and Over	M_110156526	U 27+ wks
CES102100000I	All Employees: Mining and Logging: Mining	M_123109244	Emp: mining

Table 3. Description of economic indicators