Lab 3 Predictive Modeling

Duruo Li

May 13, 2023

1 Introduction

This report is about predictive modeling. I divide the report into 3 parts: Part 1. Modeling on Census Tracts data, Part 2. Modeling on State data, and Part 3. Visualization Map.

2 Census Tracts Data

2.1 Linear Model

Questions: 1, 2, 3

I split the whole dataset into 80% training data and 20% test data. The following outcomes are all based on the test data.

According to the distribution histograms based on predictive percentage and actual percentage, see figure 1 and figure 2:

- The actual distribution is left-skewed, while the predicted distribution is relatively more symmetric.
- The range of actual range of drunk driving percentage is wider than the predicted one, i.e. actual distribution is dispersed, while predicted distribution is more centered.

Moreover, since **error** is defined as (actual value - predicted value), according to the histogram of predicted error figure 3, the majority of errors are primarily found within the range of [-10, 10], but they exhibit a left-skewed distribution. This indicates that there are more negative errors compared to positive errors. In other words, the model tends to overestimate the percentage of drunk driving incidents more frequently than it underestimates them.

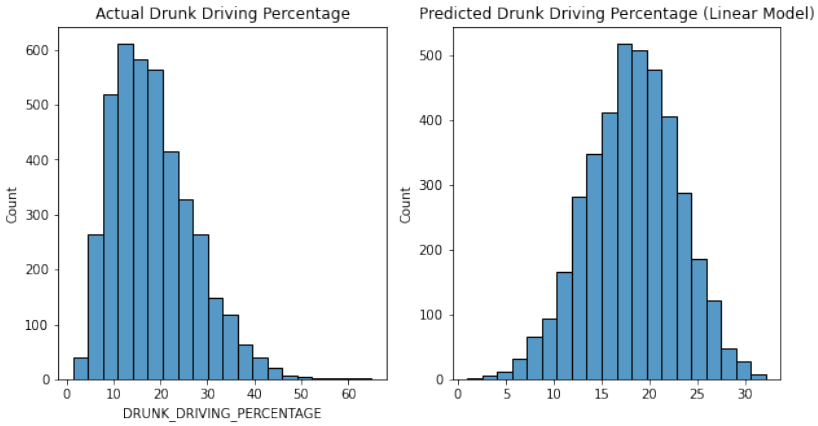

Figure 1
Histograms (separated): actual vs predicted(linear model)

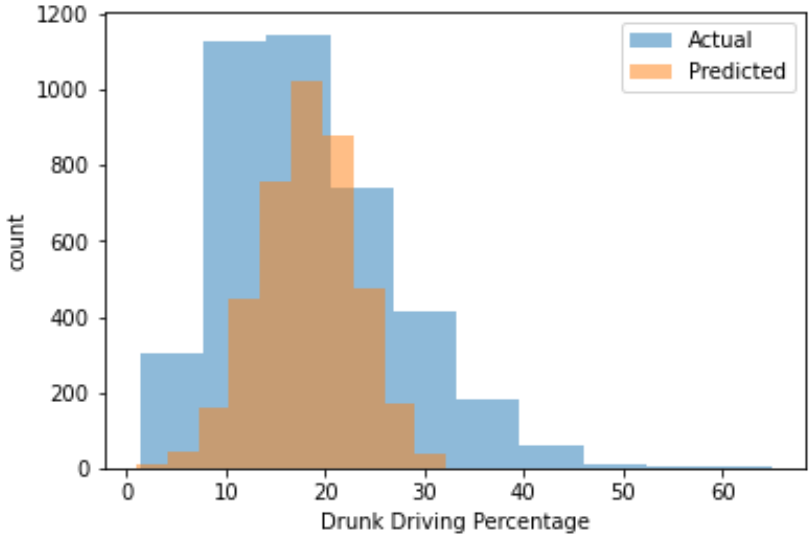

Figure 2
Histograms (overlapped): actual vs predicted(linear model)

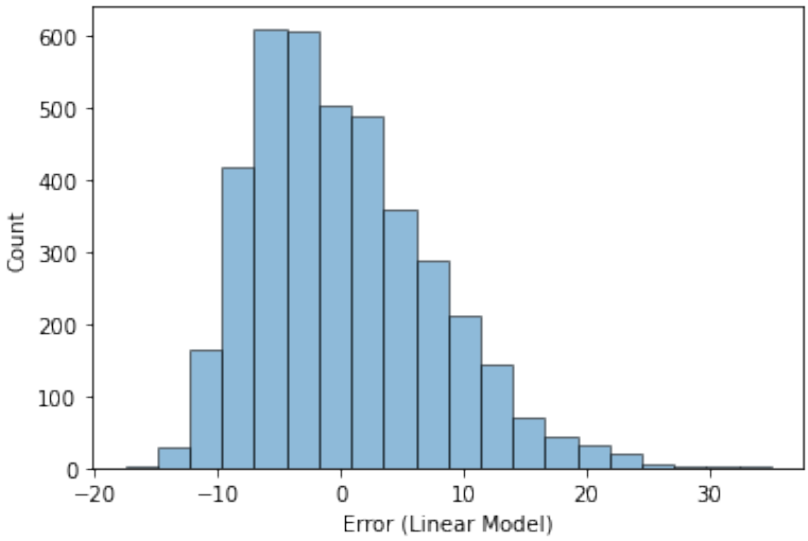

Figure 3
Histogram of errors(linear model)

Model	Test MSE
Standard model MSE	53.387
L1 model (Lasso) MSE	53.372
L2 model (Ridge) MSE	53.387

Table 1: MSE of standard and regularized linear models

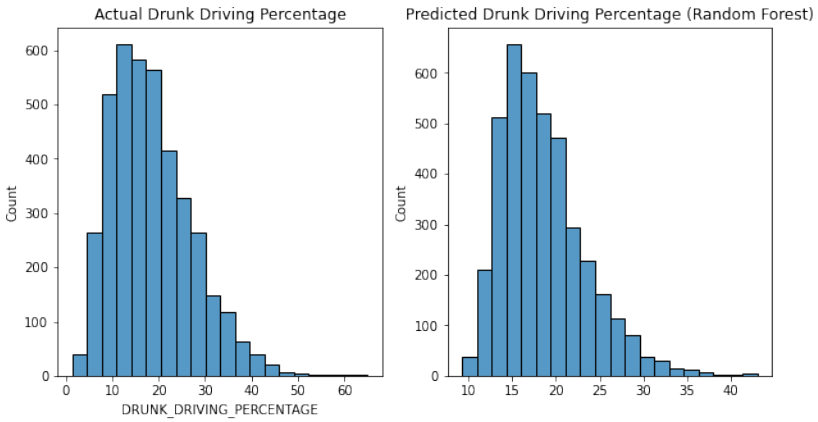

Figure 4
Histograms (separated): actual vs predicted(random forest)

Regularization:
According to the MSE table 1, we can see that regularization term doesn't seem to improve the performance since the MSE's are almost the same. During the experiment, I have used cross-validation to select best alpha, but it's weird to get quite large values, e.g. the best alpha for L1 method with 5-fold CV is 51.833, while L2 method's alpha is 70+. In the end, I choose to use alpha=0.01 which gives better performance.

2.2 Random Forest

Questions: 5, 6, 7 (it seems that there is no Q4 lol)

The MSE for random forest model is 48.015.

As for the comparisons between predicted percentages and actual percentages, see histograms Figure 4 and density plot Figure 5, both actual and predicted data are left-skewed, i.e., random forest performs better in "capturing" skewness than linear model. But actual data is still more dispersed than predicted data. In addition, the kde plot shows more clearly that predicted distribution is centered around 18.

Tune hyper-parameters:
There are many hyper-parameters which could be tuned:

- Overall: number of trees, maximum depth of tree, maximum number of leaves
- Bootstrap: the number of samples to draw from X to train a tree (data bagging size)
- Split: number of randomly selected features for split (feature bagging size); minimum number of samples required to split an internal node, minimum number of samples to be a leaf; criterion to measure split quality

I use grid search method by trying different values of several hyper-parameters, e.g. number of trees, maximum depth of trees, feature bagging size, etc. My search is based on such a group of hyperparameters:
"n_estimators": [100, 200, 500],
"max_depth": [None, 10, 20],

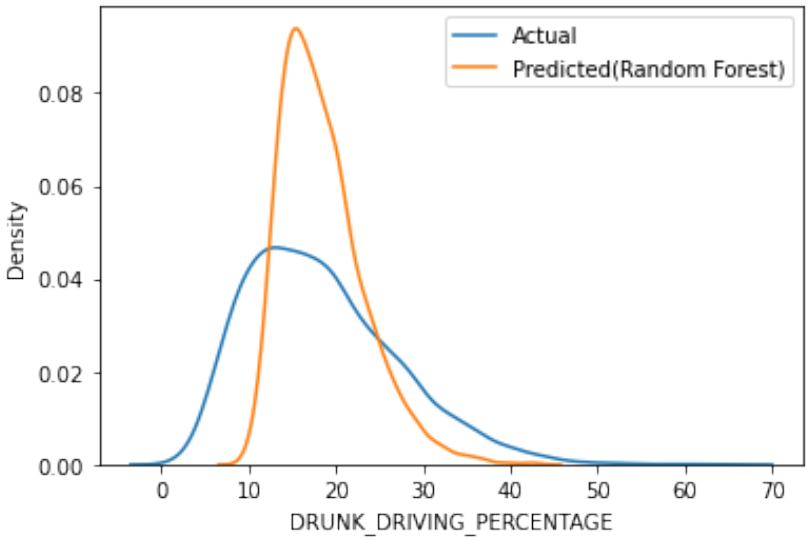

Figure 5
Density plot(overlapped): actual vs predicted(random forest)

"min_samples_split": [2, 5, 10],
"min_samples_leaf": [1, 2, 4],
"max_features": ["sqrt", "log2"]

According to the experiments, the best choice is: 'max_depth': None, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 500, and corresponding MSE=49.229, which is greater than my original model. But it inspires me that if only set n_estimators=500, and keep all the other parameters as default values, performance might be improved. It is then proved to be true, for which the MSE= 47.850.

The experiment shows that "number of trees" is a very important parameter.

2.3 Neural Networks

Questions: 8, 9, 10

After setting hyper-parameters: num_epochs=100, batch_size=200, and hidder_size=15, optimizer=Adam, I train a three-layer neural network. The MSE=50.3181.

The average value of y_test is 18.468, while the average value of the predicted y is 18.304. When observing the histogram Figure 6 and the kernel density estimation plot Figure 7, we can observe that the predicted values tend to cluster around 18. This suggests that the network has not effectively learned from the data, and its best guess is simply the mean plus a random error. To potentially address this issue, one possible solution could be to increase the model complexity, such as by adding hidden layers, in order to alleviate the problem.

However, it seems to be a consistent problem. After training some more complex networks in the future, the predicted distribution is also clustering around mean.

Tune hyper-parameters:

I use grid search to tune the neural network. The hyper parameters I choose are:

- Number of layers: 3, 4, 5, 6
- Activation function: nn.Identity, nn.ReLU, nn.ELU, nn.ReLU6, nn.GELU, nn.Softplus, nn.Softsign, nn.Tanh, nn.Sigmoid, nn.Hardsigmoid
- Optimizer: optim.SGD, optim.RMSprop, optim.Adagrad, optim.Adadelta, optim.Adam, optim.Adamax, optim.NAdam

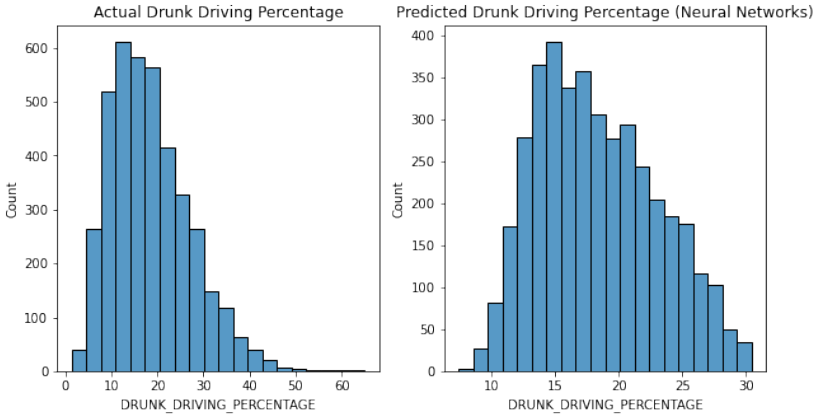

Figure 6
Histograms (separated): actual vs predicted(neural networks)

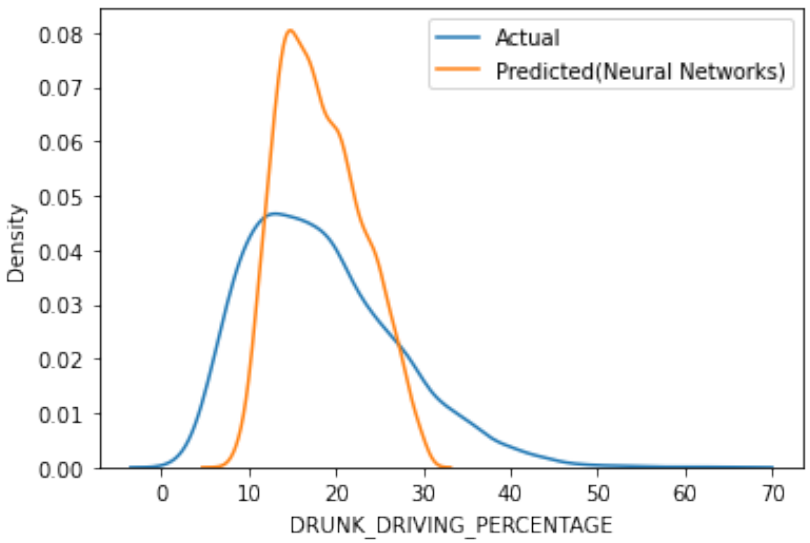

Figure 7
Density plot (overlapped): actual vs predicted(neural networks)

Model	Census Tracts	State
Linear Model	52.304	265.572
Random Forest	48.300	46.213
Neural Network	47.468	59.609

Table 2: Model performance on two datasets

The addition of a hidden layer proves to be a beneficial choice, resulting in a reduction of the MSE from 50 to approximately 45. However, despite trying different activation functions and optimizers, the MSE does not show significant improvement. Nonetheless, it is important to avoid particularly poor-performing functions, such as the Adagrad optimizer.

It turns out that the best tuning model is a 4-layer neural network with ELU activation function and SGD optimizer, whose test MSE is 45.9004 ¡ 50.3181, i.e., the network has been improved.

3 State Data

Questions: 11, 12, 13

3.1 Model comparison

To compare the performance of linear model, random forest, and neural network on census tracts data and state data, I choose to use cross-validation to calculate the mean MSE. Moreover, for neural network, since CV is usually used for selecting parameters, I instead calculate "robust" MSE (mean of 5 MSE's) by changing the random state when splitting data.

See Table 2, the results indicate that, apart from the Random Forest model, the other two models exhibit poorer performance on the state dataset compared to the census tracts dataset.

Particularly, the linear model performs notably worse. Upon examining the 5-fold cross-validation MSEs of [71.94865309, 901.84858745, 184.47215192, 97.39966436, 72.18858428], it is evident that two of the MSE values, specifically 901.85 and 184.47, are exceptionally large. These outliers significantly contribute to the overall large mean MSE.

3.2 Transfer: linear model

The prediction MSE for transferring linear model is 4396.447 while the MSE for original data is 52.185. Thus, the trained linear model doesn't seem to transfer.

3.3 Transfer: neural networks

I saved the model trained on the census dataset as model_0. I then proceeded to perform fine-tuning on model_0 using the state data, repeating the process five times with different random states for data splitting. The number of epochs is set to be 10 to avoid overfitting. Subsequently, I calculated the MSE for each fine-tuned model.

The average MSE of the transferred model is 33.469, which is significantly lower than the baseline MSE of 59.609 (Question 11). This demonstrates that the transfer of knowledge from the census dataset to the state dataset greatly improves performance.

The MSE values for the fine-tuned models are as follows: [18.113, 56.464, 33.302, 30.535, 28.931].

4 Visualization: Map

Questions: 14, 15

Figure 8 shows the percentage of drunk driving accidents in different states.

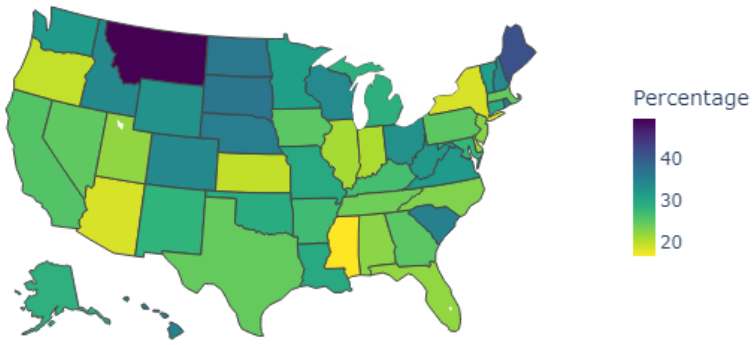

Figure 8
Choropleth map of the US States (percentage of drunk driving accidents)

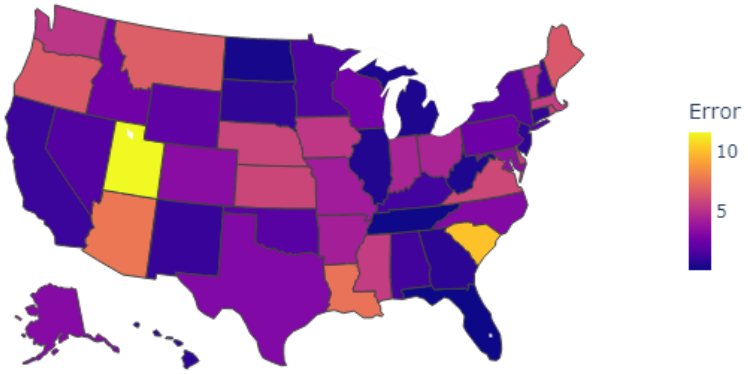

Figure 9
Choropleth map of the US States (prediction error of the neural network)

Figure 9 shows the magnitude of estimator's error for different states based on the neural network (4-layer, ELU activation function, and SGD optimizer). And Florida seems to be the best predicted state (error = 0.046015), while Utah seems to be the hardest to predict (error = 11.547348).

5 Conclusion

For prediction, neural networks model doesn't always perform the best, especially before tuning. Considering interpretation, both linear model and random forest seem to be better choices. However, when it comes to transfer learning, i.e., extracting "background information" from a larger dataset to help prediction on small datasets, neural networks perform quite well due to its natural ability in further adaptation.

In conclusion, to find the best predictive model, we need to choose carefully based on the characteristics of data and our goals.