Anderson Hsiao
CS 589

Final Project

1. Introduction

The objective of this project is to build and test several machine learning models to predict whether a passenger on the Titanic survived. We are using a dataset from Kaggle that contains information about 891 passengers on the Titanic, which includes information such as their ticket class, sex, and number of family members. In this project, we will be using this dataset to test the effectiveness of three different types of models: a kernel-based support vector model, a neural network model, and a tree-based model.

2. Methodology

2.1 Determining the best pre-processing method

First, I tested whether the model would work best using the original data, standard normalized data, or PCA normalized data. To do this, the data was split into training data and testing data at an 80:20 split.

Prior to this, initial preprocessing included reading the data from the CSV, splitting the data into inputs and outputs, and converting non-numerical values into numerical ones, which was done using the pd.get_dummies function. These methods were adopted from the Titanic Tutorial[1] notebook on Kaggle.

The kernel-based method was a Support Vector Classifier initialized with an radial basis function (rbf). The neural network was an Multi-Layer Perceptron Classifier (MLPClassifier) set with one hidden layer of size 25, regularization parameter of 0.001, and a maximum of 1,000 iterations; all other parameters were kept as default. The tree-based method was a Decision Tree Classifier set to default values.

The features used were Pclass, Sex, SibSp, and Parch, which were the features used in the Titanic Tutorial. These features did not have any empty rows, and based on my knowledge of what factors contributed to who survived the Titanic, these appeared to be good features to start off.

Using the trained model, I then applied it onto the testing split and calculated the accuracy of the predictions. For each type of pre-processing, I tested it 100 times, using different train-test splits by changing the random_state value of the sklearn train_test_split function. I calculated the average accuracy for the three different models across the ten runs. Based on test results, I decided to keep the data in its original form.

2.2 K-fold cross validation

Two of the best ways for improving the results of a neural network model is to adjust the step size and regularization (alpha in sklearn) hyperparameters. I tested combinations of 10^{-3}, 10^{-4}, … , 10^{-7} for both step size and alpha.

The neural network was once again an MLPClassifier with one hidden layer of size 25.

For each combination, I used k-fold validation, in which I split the training data into 4 parts. Every fold is used as the validation set once. When a fold is selected to be the validation set, the other folds are used to train the model. Once the model is trained, the model is used to predict values for the validation set. Once all validation sets have been tested, the average is taken across all of them. The hyperparameters that produce the best average validation accuracy are selected for the final model.

2.3 Feature engineering

Finally, I wanted to test which features were the best to include for testing. The features I used were:

- Pclass
- Sex
- SibSp
- Parch
- Fare
- Embarked

These features were selected because they all had numerical values or values that could easily be converted to numbers (i.e. only a handful of options to pick from), and because they did not have empty values for any rows.

To start, I trained all six features individually and saved the one that had the highest testing accuracy. Then, from the remaining five features, I trained each feature combined with the first feature selected from before, and determined which new feature would result in the highest training accuracy when added. This was then repeated for all other features until no features were left.

For each feature or combination of features, I trained the model 100 times using randomized train-test splits. Since all three models produce similar accuracies, the average testing accuracy across all three models was reported.

3. Results

3.1 Determining the best pre-processing method

	Original Data	**Standard Normalized**	**PCA Normalized**
Kernel	0.803	0.803	0.799
Neural Network	0.799	0.797	0.801
Tree	0.799	0.797	0.801

The table above shows the accuracy of the different types of pre-processing on the three different models. The difference between them is very small, with original data being slightly better than the others. Although the exact accuracies could vary depending on how the train-test splits are randomized, I decided to go with the original data as it slightly decreases the running time required to normalize the data.

3.2 K-fold cross validation

K-fold validation determined that the best combination of hyperparameters was 0.001 for both the learning rate and regularization parameter. It produced an average validation accuracy of 0.816. When these values were applied for the original training and testing data splits, it resulted in an accuracy of 0.771 on the test set, which is around the same range.

3.3 Feature engineering

Number of features	Best features determined	Average accuracy across all 3 models
1	Sex	0.788
2	Sex, SibSp	0.796
3	Sex, SibSp, Parch	0.797
4	Sex, SibSp, Parch, Embarked	0.801
5	Sex, SibSp, Parch, Embarked, Pclass	0.803
6	Sex, SibSp, Parch, Embarked, Pclass, Fare	0.800

Based on the results, sex was the most influential feature in determining whether a passenger survived or not, achieving 78.8% accuracy on its own. Adding additional features to the models improved accuracy, but not by much. It appears that sex is the strongest determinant of survival, while the other features probably improved accuracy for just a few special cases. In addition, adding more features doesn't always lead to better results, as we can see from the accuracy going down from 80.3% for 5 features to 80.0% for 6 features.

4. Conclusion

There appears to be very minimal differences in the accuracy of three models when trained with different pre-processing methods, different hyperparameters, and different input features. It also appears that sex is the most signficant feature that determined whether or not a passenger survived, with it being to correctly identify close to 80% of passengers when used as a standalone training feature.

Potential areas for improvement could include training models with more features, getting a dataset with more information, using different models or even libraries, and maybe custom-building a model specifically for this task.

5. References

1. https://www.kaggle.com/code/alexisbcook/titanic-tutorial