IBM Machine Learning

Titanic Survival Prediction

Yu Hsuan HSIEH

April 2023

Summary & Objective

This is a project analyzing Titanic Passenger Survivals.

The dataset used in this project is provided by Kaggle.

We'll first observe the dataset, cleaning will be performed if needed, then fit the dataset into 3 different sets of variables for a CNN model.

Models will be evaluated and compared to find out the best model that fits the data.

The main objective of this analysis will be trying to train a model that best predicts the survival of passengers on Titanic with given information.

Data Observation

Dataset

From the result of data.head() and data.info(), we can see that there're 12 columns in the dataset. Each column is integer, float or string, thus data cleaning is required.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  891 non-null    int64
 1   Survived     891 non-null    int64
 2   Pclass       891 non-null    int64
 3   Name         891 non-null    object
 4   Sex          891 non-null    object
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64
 7   Parch        891 non-null    int64
 8   Ticket       891 non-null    object
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object
 11  Embarked     889 non-null    object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

Getting useful columns

Among all the columns, the 'PassengerId', 'Name', 'Fare', 'Ticket', and 'Cabin' columns don’t seen to affect the result of whether the passenger survived or not, thus we’ll remove these columns from our dataset, leaving 6 columns.

Also, we removed rows with NA, leaving 712 rows.

	Pclass	Sex	Age	SibSp	Parch	Embarked
0	3	male	22.0	1	0	S
1	1	female	38.0	1	0	C
2	3	female	26.0	0	0	S
3	1	female	35.0	1	0	S
4	3	male	35.0	0	0	S
...	...	...	...	...	...	...
885	3	female	39.0	0	5	Q
886	2	male	27.0	0	0	S
887	1	female	19.0	0	0	S
889	1	male	26.0	0	0	C
890	3	male	32.0	0	0	Q

712 rows × 6 columns

Encoding & Scaling

We must encode columns with strings, namely ‘Sex’ and ‘Embarked’ using LabelEncoder().

	Pclass	Sex	Age	SibSp	Parch	Embarked
0	3	1	22.0	1	0	2
1	1	0	38.0	1	0	0
2	3	0	26.0	0	0	2
3	1	0	35.0	1	0	2
4	3	1	35.0	0	0	2
...	...	...	...	...	...	...
885	3	0	39.0	0	5	1
886	2	1	27.0	0	0	2
887	1	0	19.0	0	0	2
889	1	1	26.0	0	0	0
890	3	1	32.0	0	0	1

712 rows × 6 columns

We will also scale the columns in the dataset using MinMaxScaler().

	Pclass	Sex	Age	SibSp	Parch	Embarked
0	1.0	1	0.271174	0.2	0.000000	1.0
1	0.0	0	0.472229	0.2	0.000000	0.0
2	1.0	0	0.321438	0.0	0.000000	1.0
3	0.0	0	0.434531	0.2	0.000000	1.0
4	1.0	1	0.434531	0.0	0.000000	1.0
...	...	...	...	...	...	...
885	1.0	0	0.484795	0.0	0.833333	0.5
886	0.5	1	0.334004	0.0	0.000000	1.0
887	0.0	0	0.233476	0.0	0.000000	1.0
889	0.0	1	0.321438	0.0	0.000000	0.0
890	1.0	1	0.396833	0.0	0.000000	0.5

712 rows × 6 columns

Conventional Neural Network (CNN)

CNN Model - Basic Settings

For the basic CNN model, we will be using the following code.

```
model = Sequential(
    [
        Dense(9, input_shape=(6,), activation='relu'),
        Dense(15, activation='relu'),
        Dense(50, activation='relu'),
        Dense(2, activation="softmax"),
    ]
)
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 9)                 63

 dense_1 (Dense)             (None, 15)                150

 dense_2 (Dense)             (None, 50)                800

 dense_3 (Dense)             (None, 2)                 102

=================================================================
Total params: 1,115
Trainable params: 1,115
Non-trainable params: 0
_________________________________________________________________
```

Settings

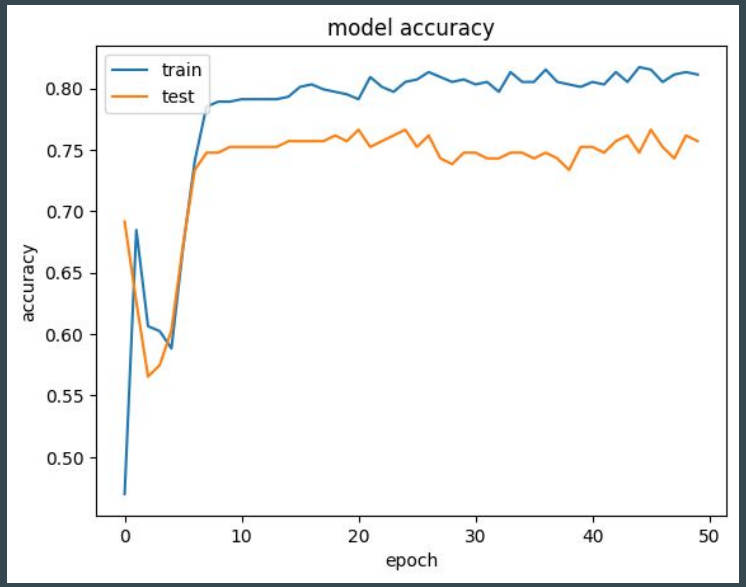
The codes for compiling and fitting are shown below.

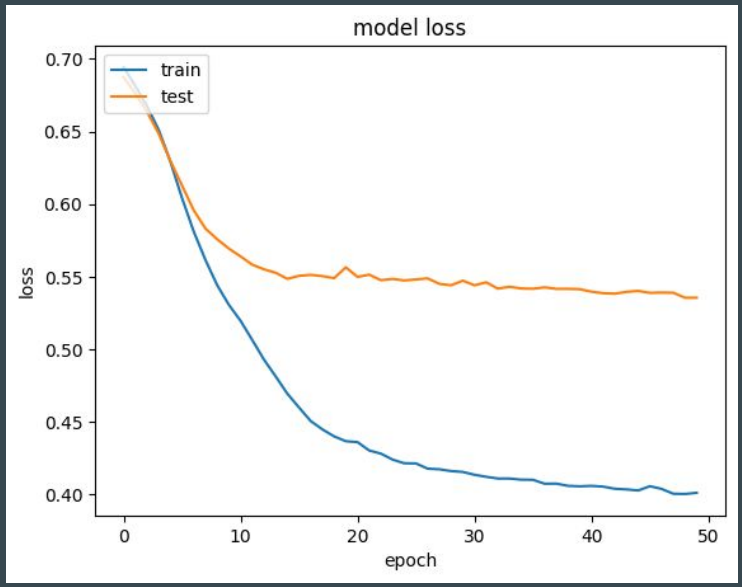
```
model.compile(loss='binary_crossentropy',
              optimizer='Adam',
              metrics=['accuracy'])

history = model.fit(X_train, y_train,
                    epochs=50, batch_size=32,
                    validation_data=(X_test, y_test),
                    verbose=1, validation_split=0.4,
                    shuffle=True)
```

CNN Model I - Results

Overfitting occurred

```
Test loss: 0.5355746746063232

Test accuracy: 0.7570093274116516
```





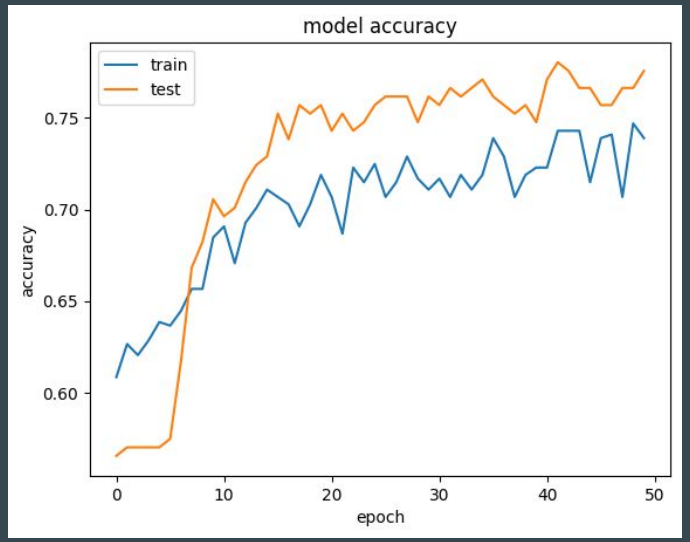
CNN Model II

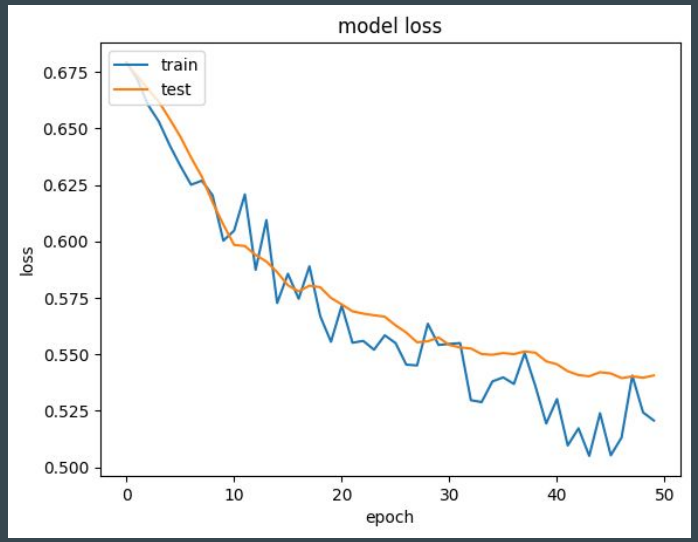
For the second model of CNN, we will try to add a dropout layer to prevent overfitting.

```
model_1 = Sequential(
    [
        Dense(9, input_shape=(6,), activation='relu'),
        Dropout(0.5),
        Dense(15, activation='relu'),
        Dropout(0.5),
        Dense(50, activation='relu'),
        Dense(2, activation="softmax"),
    ]
)
```

CNN Model II - Results

```
Test loss: 0.5406837463378906

Test accuracy: 0.7757009267807007
```





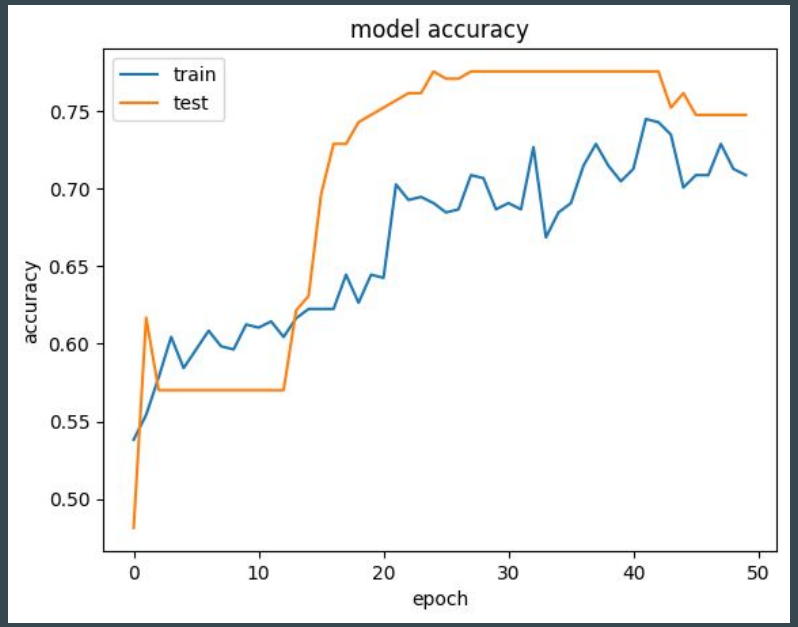
CNN Model III

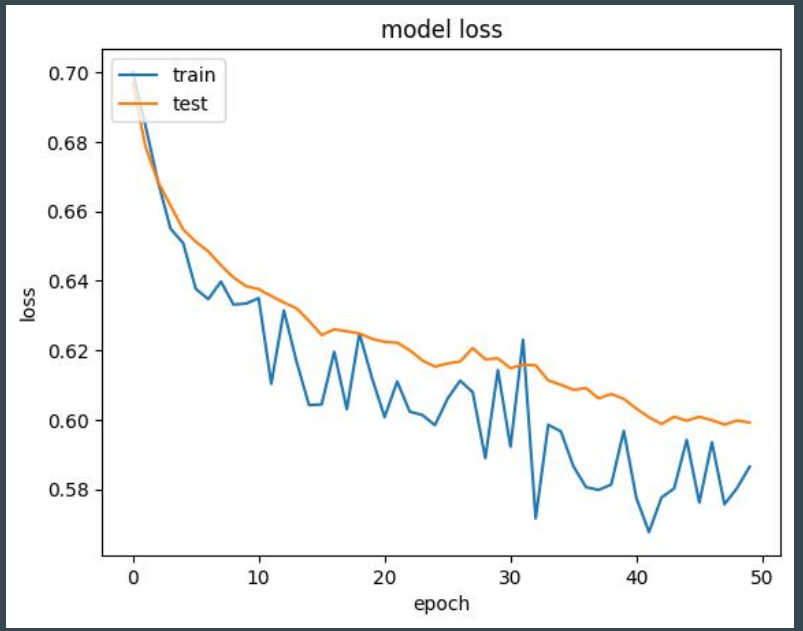
For the second model of CNN, we will try to add more dropout layers to see whether the result improved.

```
model_3 = Sequential(
    [
        Dense(9, input_shape=(6,), activation='relu'),
        Dropout(0.5),
        Dense(15, activation='relu'),
        Dropout(0.5),
        Dense(50, activation='relu'),
        Dropout(0.5),
        Dense(2, activation="softmax"),
    ]
)
```

CNN Model III - Results

```
Test loss: 0.5406837463378906

Test accuracy: 0.7616822719573975
```





Results

Comparision

	Model I	Model II	Model III
Loss	**0.536**	0.541	0.541
Accuracy	0.757	**0.776**	0.762

From table above, Model I has the lowest lost, however, Model II has the highest accuracy.
Moreover, from the history chart, we can see that Model I has severe overfitting issue, Model II and Model III improved the overfitting issue.

Possible Flaw

There are several potential flaws that can arise in CNN (Convolutional Neural Network) model testing experiments, including:

- Overfitting: CNN models, like other deep learning models, can be prone to overfitting, where the model performs well on the training data but fails to generalize to unseen data.
- Limited sample size: CNN models often require a large amount of training data to achieve good performance. In this project, only a limited number of 712 rows are used.
- Hyperparameter tuning: CNN models typically have several hyperparameters, such as learning rate, batch size, and model architecture, that need to be tuned during training. If hyperparameter tuning is not properly performed during testing, the reported performance may not reflect the optimal performance of the model.

Suggestion

For further research, we can try tuning the parameters for the CNN model, try to find the optimal parameters. Moreover, we can try evaluating the methods with more ways and deep learning models, to get a more comprehensive analysis.

Conclusion

Our study demonstrates the effectiveness of different hyperperameters for a CNN model for predicting the result of the dataset.

Overall, our findings provide useful insights for anyone interested in using deep learning methods for the Titanic dataset, and highlight the potential for further research in this area.

Thank you!