Project1 report

NYCU Spr2023: AI Capstone

Author: 110652019 林楷傑

1. Image Datasets

Description

Data Source (https://www.cs.toronto.edu/~kriz/cifar.html)
The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images.

Data Preprocessing

- Data Transformer(To augmenting data)
  - Random Rotation
  - Random Horizontal Flip
- Show some sample from the dataset

- Train test split
  - CIFAR10 has helped us accomplish this task
  - We also don't do cross validation here.
- Batch size is 128

Algorithms

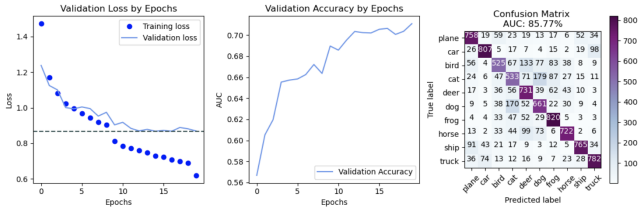
Convolutional Neural Network

The convolutional layers apply filters to the input image to identify patterns and features, while the pooling layers reduce the dimensionality of the output.

- Architecture:
  Five CNN layers with ReLU and Maxpooling layers.
- Hyperparameters and optimizer

```
epochs = 20
lr = 3e-4
weight_decay = 1e-5
gamma = 0.5
step_size = 10
optimizer = torch.optim.Adam
scheduler = torch.optim.lr_scheduler.StepLR
```

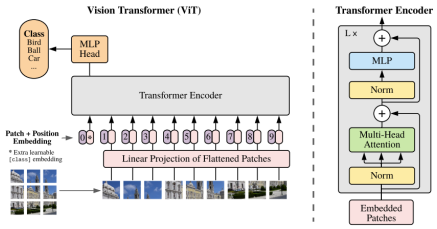
- Training
  - Training 20 epochs with 6 minutes.
  - Last epoch: val_loss: 0.8691, val_acc: 0.7108
  - Model AUC 85.77%, Accuracy 71.04% on Test Data



VISION TRANSFORMER

Vision Transformer(ViT) is a attention-based transformer architecture. ViT attains excellent results compared to state-of-the-art convolutional networks while requiring substantially fewer computational resources to train.

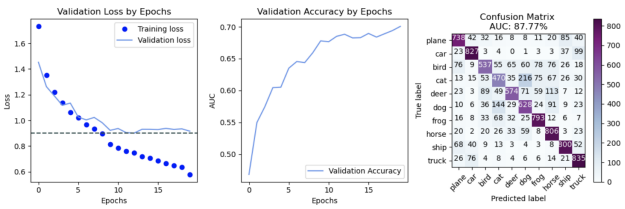
- Architecture



- Hyperparameters and optimizer

```
epochs = 20
lr = 3e-4
weight_decay = 1e-5
gamma = 0.5
step_size = 10
optimizer = torch.optim.Adam
scheduler = torch.optim.lr_scheduler.StepLR
```

- Training
  - Training 20 epochs with 1 hour.
  - Last epoch: val_loss: 0.9160, val_acc: 0.7007
  - Model AUC 87.77%, Accuracy 70.08% on Test Data



Analysis

- Compare the result when using half of training data.
  - The training time is cost half of original.
  - CNN

```
Model AUC 86.11%, Accuracy 68.21% on Test Data

              precision    recall  f1-score   support

       plane       0.73      0.71      0.72      1000
         car       0.84      0.78      0.81      1000
        bird       0.51      0.63      0.56      1000
         cat       0.50      0.53      0.51      1000
        deer       0.64      0.60      0.62      1000
         dog       0.67      0.50      0.57      1000
        frog       0.71      0.77      0.74      1000
       horse       0.71      0.75      0.73      1000
        ship       0.76      0.80      0.78      1000
       truck       0.79      0.77      0.78      1000

    accuracy                           0.68     10000
   macro avg       0.69      0.68      0.68     10000
weighted avg       0.69      0.68      0.68     10000
```

  - ViT

```
Model AUC 83.27%, Accuracy 65.39% on Test Data

              precision    recall  f1-score   support

       plane       0.71      0.64      0.68      1000
         car       0.69      0.84      0.76      1000
        bird       0.62      0.47      0.54      1000
         cat       0.52      0.37      0.43      1000
        deer       0.64      0.57      0.60      1000
         dog       0.54      0.60      0.57      1000
        frog       0.69      0.79      0.74      1000
       horse       0.66      0.75      0.70      1000
        ship       0.72      0.73      0.73      1000
       truck       0.68      0.77      0.72      1000

    accuracy                           0.65     10000
   macro avg       0.65      0.65      0.65     10000
weighted avg       0.65      0.65      0.65     10000
```

- Compare the results when using different classifiers.
  - CNN Performance

```
Model AUC 85.77%, Accuracy 71.04% on Test Data

              precision    recall  f1-score   support

       plane       0.73      0.76      0.75      1000
         car       0.83      0.81      0.82      1000
        bird       0.65      0.53      0.58      1000
         cat       0.54      0.53      0.54      1000
        deer       0.61      0.73      0.67      1000
         dog       0.61      0.66      0.63      1000
        frog       0.73      0.82      0.77      1000
       horse       0.80      0.72      0.76      1000
        ship       0.84      0.77      0.80      1000
       truck       0.79      0.78      0.79      1000

    accuracy                           0.71     10000
   macro avg       0.71      0.71      0.71     10000
weighted avg       0.71      0.71      0.71     10000
```

  - ViT Performance

```
Model AUC 87.77%, Accuracy 70.08% on Test Data

              precision    recall  f1-score   support

       plane       0.73      0.74      0.73      1000
         car       0.80      0.83      0.82      1000
        bird       0.66      0.54      0.59      1000
         cat       0.55      0.47      0.51      1000
        deer       0.73      0.57      0.64      1000
         dog       0.58      0.63      0.60      1000
        frog       0.75      0.79      0.77      1000
       horse       0.67      0.81      0.73      1000
        ship       0.78      0.80      0.79      1000
       truck       0.73      0.83      0.78      1000

    accuracy                           0.70     10000
   macro avg       0.70      0.70      0.70     10000
weighted avg       0.70      0.70      0.70     10000
```

- Compare the results when using different settings and/or hyper-parameters for the same classifier. I change learning rate to 1e-3, weight_decay to 1e-6, gamma to 0.5 and step_size to 5.

- CNN2 Performance

```
Model AUC 87.77%, Accuracy 70.08% on Test Data

              precision    recall  f1-score   support

       plane       0.73      0.74      0.73      1000
         car       0.80      0.83      0.82      1000
        bird       0.66      0.54      0.59      1000
         cat       0.55      0.47      0.51      1000
        deer       0.73      0.57      0.64      1000
         dog       0.58      0.63      0.60      1000
        frog       0.75      0.79      0.77      1000
       horse       0.67      0.81      0.73      1000
        ship       0.78      0.80      0.79      1000
       truck       0.73      0.83      0.78      1000

    accuracy                           0.70     10000
   macro avg       0.70      0.70      0.70     10000
weighted avg       0.70      0.70      0.70     10000
```

- ViT2 Performance

```
Model AUC 87.77%, Accuracy 70.08% on Test Data

              precision    recall  f1-score   support

       plane       0.73      0.74      0.73      1000
         car       0.80      0.83      0.82      1000
        bird       0.66      0.54      0.59      1000
         cat       0.55      0.47      0.51      1000
        deer       0.73      0.57      0.64      1000
         dog       0.58      0.63      0.60      1000
        frog       0.75      0.79      0.77      1000
       horse       0.67      0.81      0.73      1000
        ship       0.78      0.80      0.79      1000
       truck       0.73      0.83      0.78      1000

    accuracy                           0.70     10000
   macro avg       0.70      0.70      0.70     10000
weighted avg       0.70      0.70      0.70     10000
```

- It seens that the performance would not vary a lot.

- Compare with results given in the respective websites or literature if they are available.
  - Since CIFAR10 is a classic image data source. Many people would expriment with it.
  - **CNN-based(VGG16)** (https://huggingface.co/edadaltocg/vgg16_bn_cifar10)
  - **Transformer based(ViT)** (https://huggingface.co/aaraki/vit-base-patch16-224-in21k-finetuned-cifar10)
  - Since my computing resource is not enough, I can't reach same high performance like others did. But this experiment let me gain many experiments about ML and DL algorithms.

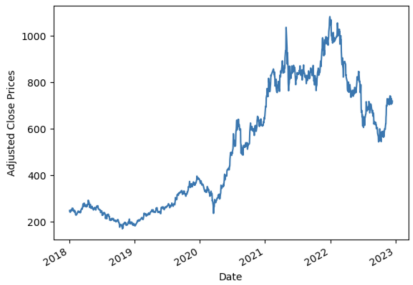
2. Stock price prediction

Description

Data Source (https://tw.finance.yahoo.com/)

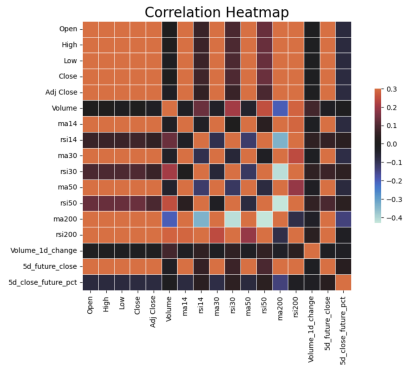
yfinance API (https://pypi.org/project/yfinance/)

- yfinance offers a threaded and Pythonic way to download market data from Yahoo!R finance.
- Download the stock data of MediaTek(聯發科) from 2018 to 2022 end.



Data preprocessing

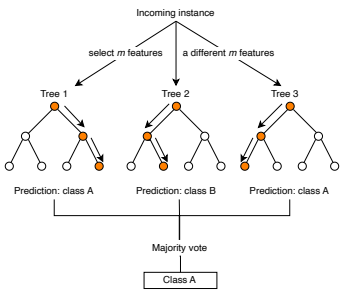
- Feature Engineering
  - Compute the MA value and RMI value
- Target is 5 day future close.
- Correlation Heatmap



Algorithms

Random Forest Regressor

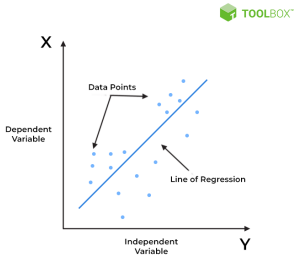
- **Use sklearn library** (https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html)
- A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting.



Linear Regression

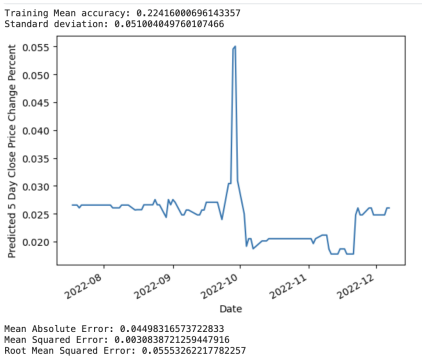
- **Use sklearn library** (https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html)
- LinearRegression fits a linear model with coefficients w = (w1, ..., wp) to minimize the residual sum of squares between the observed targets in the dataset, and the

targets predicted by the linear approximation.

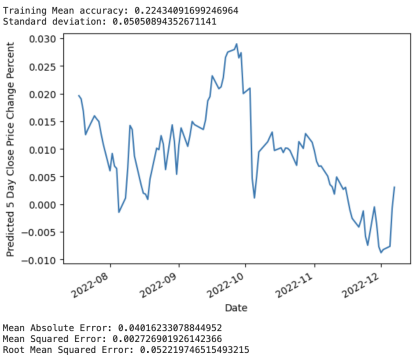


Analysis

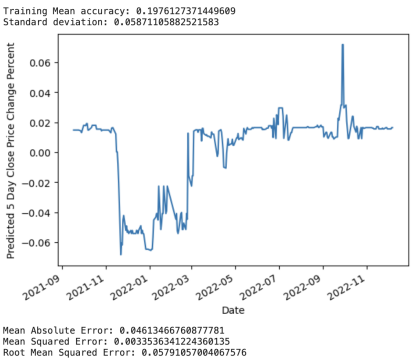
- Training evaluation with Cross Validation
  - Since the data is continuous, I did not use confusion matrix here.
  - **Cross Validation** (https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.KFold.html)
  - Random Forest:



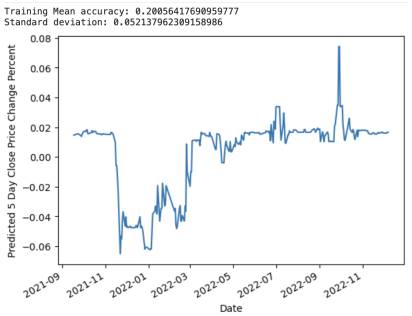
  - Linear Regression:



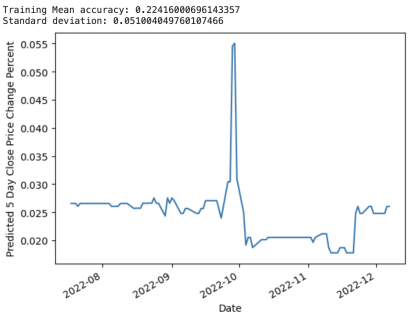
- Compare the results when using different amounts of training data.
  - Train size = 70% (Other experiments are 90%)
  - Random Forest



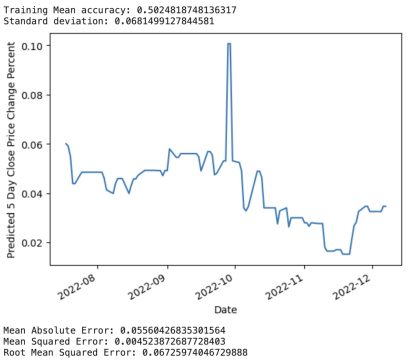
- Linear Regression



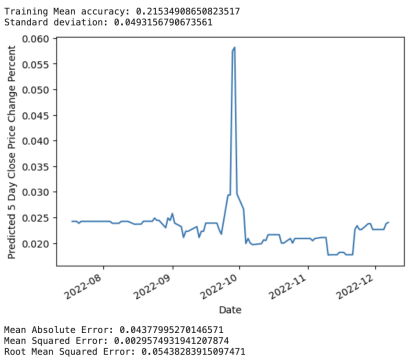
- Compare the results when using different settings and/or hyper-parameters for the same classifier.
  - Tune different hyper parameter of random forest
  - (n_estimators, max_depth) = (50, 4)



  - (n_estimators, max_depth) = (20, 8)



  - (n_estimators, max_depth) = (100, 4)



3. Music Classification

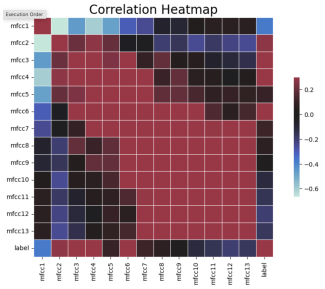
Description

- I dowload some EDM from Soundcloud and classify them into two category, which are house music and non-house

music.

Data preprocessing

- Feature extraction
  - Use librosa python library to extract the mfcc features.The MFCC feature extraction technique basically includes windowing the signal, applying the DFT, taking the log of the magnitude, and then warping the frequencies on a Mel scale, followed by applying the inverse DCT.
  - **librosa** (https://librosa.org/doc/latest/index.html)
- Correlation Heatmap
  - We can find that many features has positive correlation with label.



Algorithms

Decision Tree Classifier

- **Use sklearn library** (https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier)
- A decision tree is a non-parametric supervised learning algorithm, which is utilized for both classification and regression tasks. It has a hierarchical, tree structure, which consists of a root node, branches, internal nodes and leaf nodes.

K_Neighbors Classifier

- **Use sklearn library** (https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html)
- The k-nearest neighbors algorithm, also known as KNN or k-NN, is a non-parametric, supervised learning classifier, which uses proximity to make classifications or predictions about the grouping of an individual data point.

Analysis

- Training Evaluation and compare with different model. Using 80% training data (cross validation/precision/recall/F1/AUROC)
- kf = KFold(n_splits=10, shuffle=True, random_state=42)
  - Decision Tree(max_depth=5)

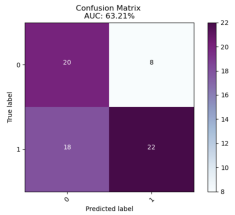
```
Training Mean accuracy: 0.6851851851851851
Standard deviation: 0.08156561313164903

 Model AUC 63.21%, Accuracy 61.76% on Test Data

              precision    recall  f1-score   support

           0       0.53      0.71      0.61        28
           1       0.73      0.55      0.63        40

    accuracy                           0.62        68
   macro avg       0.63      0.63      0.62        68
weighted avg       0.65      0.62      0.62        68
```



  - KNN(n_neighbors=1)

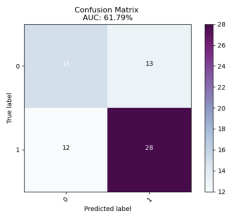
```
Training Mean accuracy: 0.7148148148148149
Standard deviation: 0.06425685767739805

 Model AUC 61.79%, Accuracy 63.24% on Test Data

              precision    recall  f1-score   support

           0       0.56      0.54      0.55        28
           1       0.68      0.70      0.69        40

    accuracy                           0.63        68
   macro avg       0.62      0.62      0.62        68
weighted avg       0.63      0.63      0.63        68
```



- Compare the results when using 70% training data.(Use Decision Tree to illustrate.)
  - Decision Tree(max_depth=5)

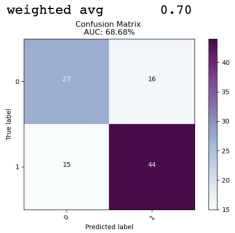
```
Training Mean accuracy: 0.6730072463768115
Standard deviation: 0.10171500524276342

 Model AUC 68.68%, Accuracy 69.61% on Test Data

              precision    recall  f1-score   support

           0       0.64      0.63      0.64        43
           1       0.73      0.75      0.74        59

    accuracy                           0.70       102
   macro avg       0.69      0.69      0.69       102
weighted avg       0.70      0.70      0.70       102
```



- Compare the results when using different settings and/or hyper-parameters for the same classifier.

- Decision Tree(max_depth=10)

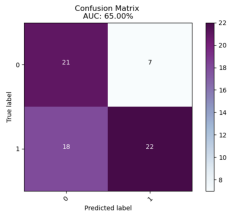
```
Training Mean accuracy: 0.7222222222222222
Standard deviation: 0.06879324304076558

 Model AUC 65.00%, Accuracy 63.24% on Test Data

              precision    recall  f1-score   support

           0       0.54      0.75      0.63        28
           1       0.76      0.55      0.64        40

    accuracy                           0.63        68
   macro avg       0.65      0.65      0.63        68
weighted avg       0.67      0.63      0.63        68
```



- KNN(n_neighbors=10)

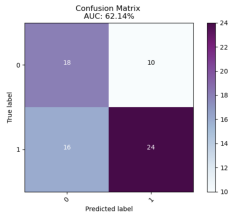
```
Training Mean accuracy: 0.6925925925925925
Standard deviation: 0.09079741238615749

 Model AUC 62.14%, Accuracy 61.76% on Test Data

              precision    recall  f1-score   support

           0       0.53      0.64      0.58        28
           1       0.71      0.60      0.65        40

    accuracy                           0.62        68
   macro avg       0.62      0.62      0.61        68
weighted avg       0.63      0.62      0.62        68
```



Discussion

- In these experiments. I used CNN and ViT to predict image, random forest and linear regression to predict stock and decision tree and KNN to classify music genre. The result is in my expect since I did not make many improvement to performance.
- I think although the algorithm we use is import, data preparing and feature engineering are also very important, too. Good performance models are based on good quality data.
- If I have more time to do this project, I would use bigger model and higher quality data to train. I want to train the difference between ResNet and Vision Transfomer with more layers. And combine the CNN with transformers to reach better performance.
- This is my first to hand write the transformer architecture, which is pretty interesting. I have learned that collecting data would be exhauting. In the future, I would have appreciate with the data source.