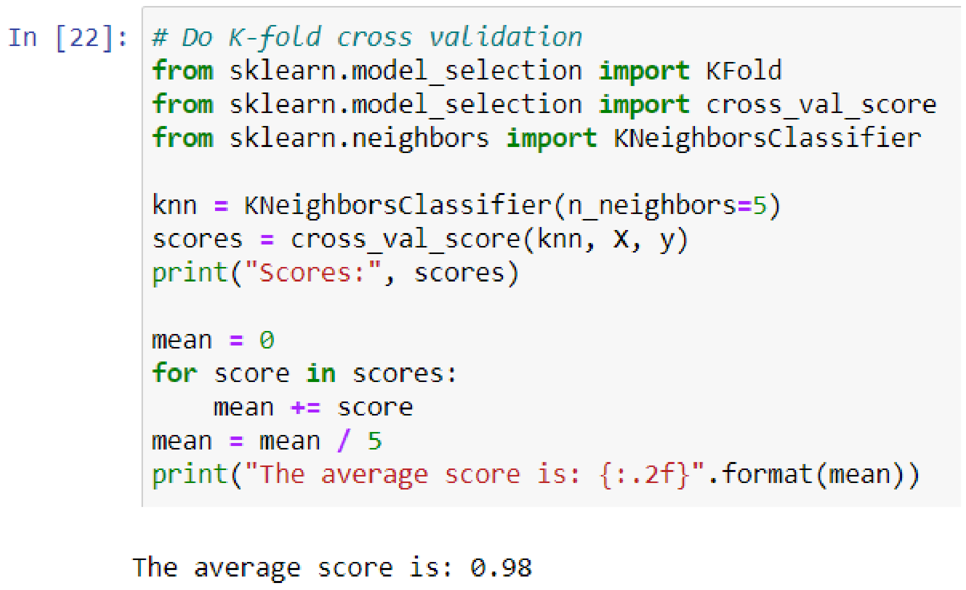
Lab1 Report

309552043 楊宇正

1. Do K-fold cross validation

When I firstly trying KFold function directly, it returns some KeyError due to some of the indexes do not exist. (I guess the reason may be that I dropped duplicated entries when invoking “**read_csv**” function which makes the index not continuous.)
Thus, I somehow found that there’s another function called **“cross_val_score”** which will apply the K-Fold cross validation internally and return a list with scores for each split inside.
The following figure shows the average score after doing K-Fold CV where K=5 (default). The avg. score is 0.98.



```
In [22]: # Do K-fold cross validation
         from sklearn.model_selection import KFold
         from sklearn.model_selection import cross_val_score
         from sklearn.neighbors import KNeighborsClassifier

         knn = KNeighborsClassifier(n_neighbors=5)
         scores = cross_val_score(knn, X, y)
         print("Scores:", scores)

         mean = 0
         for score in scores:
             mean += score
         mean = mean / 5
         print("The average score is: {:.2f}".format(mean))

         The average score is: 0.98
```

2. Data Processing

For data preprocessing, there’s a lot of works need to be done.
I firstly make a feature_names list manually since the original data doesn’t have column names.
Then, I set some parameters to drop some invalid or duplicate entries when importing the dataset and transform it to a Pandas.dataframe structure.
Finally, I assigned the feature_name list to it’s columns

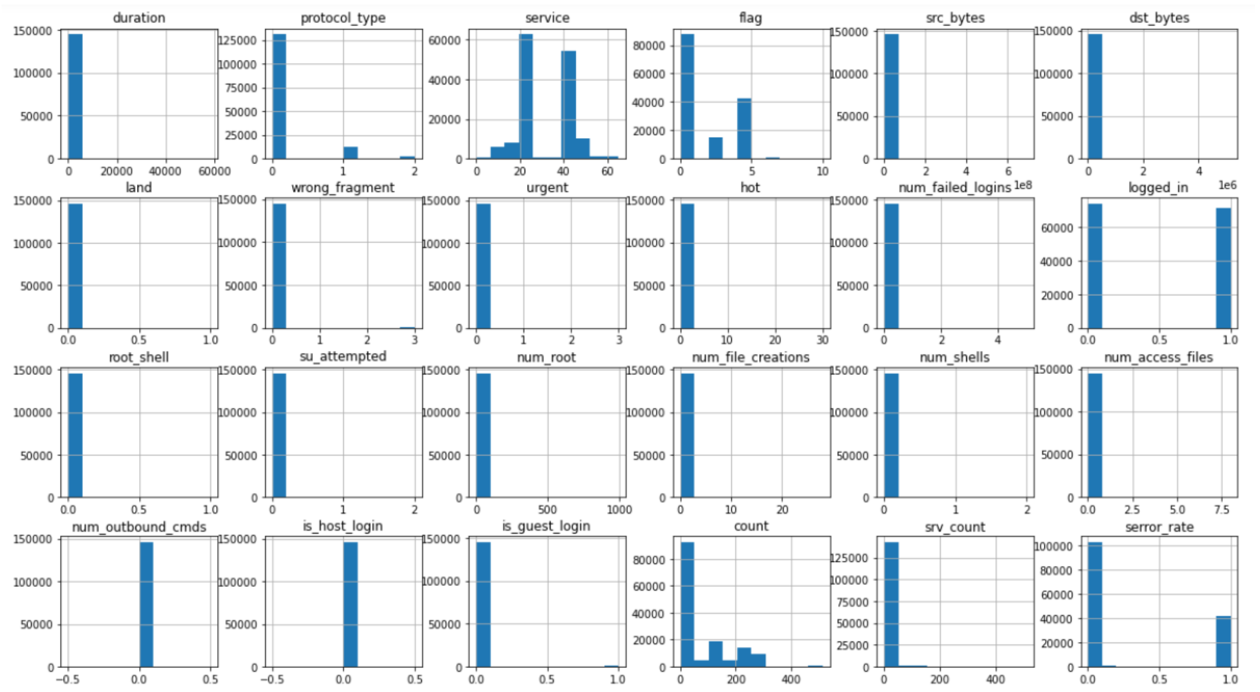
```python
feature_names = ['duration', 'protocol_type', 'service', 'flag', 'src_bytes', 'dst_bytes', 'land',
       'wrong_fragment', 'urgent', 'hot', 'num_failed_logins', 'logged_in',
       'num_compromised', 'root_shell', 'su_attempted', 'num_root',
       'num_file_creations', 'num_shells', 'num_access_files',
       'num_outbound_cmds', 'is_host_login', 'is_guest_login', 'count',
       'srv_count', 'serror_rate', 'srv_serror_rate', 'rerror_rate',
       'srv_rerror_rate', 'same_srv_rate', 'diff_srv_rate',
       'srv_diff_host_rate', 'dst_host_count', 'dst_host_srv_count',
       'dst_host_same_srv_rate', 'dst_host_diff_srv_rate',
       'dst_host_same_src_port_rate', 'dst_host_srv_diff_host_rate',
       'dst_host_serror_rate', 'dst_host_srv_serror_rate',
       'dst_host_rerror_rate', 'dst_host_srv_rerror_rate', 'target']

# import the training dataset and transform it to a dataframe
df = pd.read_csv('kddcup.data_10_percent.txt', error_bad_lines=False, low_memory=False).drop_duplicates()
df.columns = feature_names
```

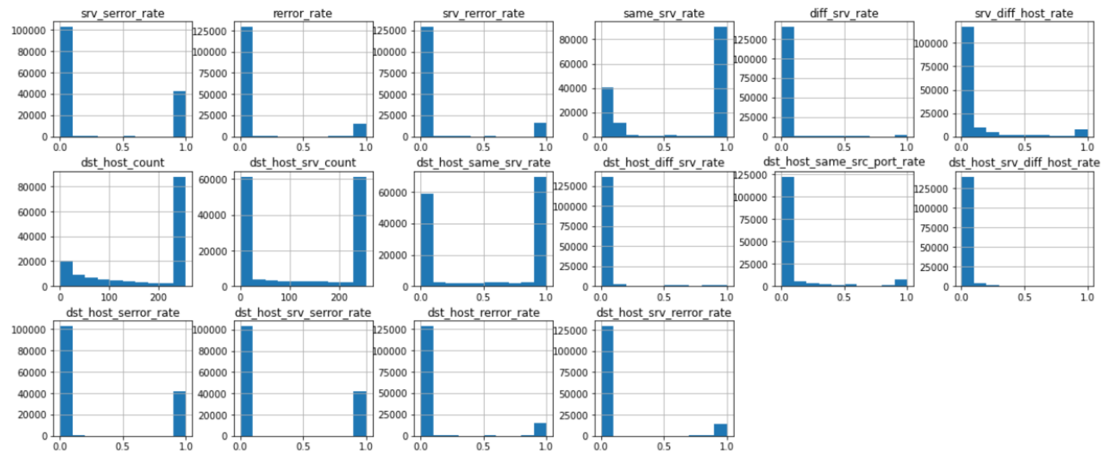
Same concepts and jobs for test set.

3. Visualize Data

For data visualization, I use the **hist()** function from Pandas.DataFrame to plot histograms for each feature in training data to see the overall distribution of data.

```python
# Visualize the data
import matplotlib.pyplot as plt
X_feature_names = ['duration', 'protocol_type', 'service', 'flag', 'src_bytes', 'dst_bytes', 'land',
       'wrong_fragment', 'urgent', 'hot', 'num_failed_logins', 'logged_in', 'root_shell',
       'su_attempted', 'num_root',
       'num_file_creations', 'num_shells', 'num_access_files',
       'num_outbound_cmds', 'is_host_login', 'is_guest_login', 'count',
       'srv_count', 'serror_rate', 'srv_serror_rate', 'rerror_rate',
       'srv_rerror_rate', 'same_srv_rate', 'diff_srv_rate',
       'srv_diff_host_rate', 'dst_host_count', 'dst_host_srv_count',
       'dst_host_same_srv_rate', 'dst_host_diff_srv_rate',
       'dst_host_same_src_port_rate', 'dst_host_srv_diff_host_rate',
       'dst_host_serror_rate', 'dst_host_srv_serror_rate',
       'dst_host_rerror_rate', 'dst_host_srv_rerror_rate']
X[X_feature_names].hist(figsize=(20,20))
```





4. Feature Transformation

For feature transformation, I transformed the feature **“service”** from *str* to *int*.
Since the fit() function we’re going to invoke latter doesn’t allow the str format so we have to convert it to numerical values.
The function I used here is called **“fit_tranform”** from **LabelEncoder** class.

```
# Since the values in 'service' is str, which is an "object" dtype and
# need to be converted to numerical values before invoking fit() function
le = LabelEncoder()
if X['service'].dtype == object:
    X['service'] = le.fit_transform(X['service'])
```

While some features such as “protocol_type” and “flag”, I did a manually mapping to convert them from *str* to *int*.

```
# mapping feature_name string to index numbers
X['protocol_type'] = X['protocol_type'].replace({ 'tcp': 0, 'udp': 1, 'icmp': 2 })
#X['service'] = label_binarize(X['service'], classes=[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20,
X['flag'] = X['flag'].replace({ 'SF': 0,'S1': 1,'REJ': 2,'S2': 3,'S0': 4,'S3': 5,'RSTO': 6,'RSTR': 7,'RSTOS0': 8,'OTH': 9,'SH':
```

5. Feature Selection

I've check the website to see the description of each feature, I think all of them may be useful for identifying an attack type. Thus, I don't know how to choose them, to be honest. Since I still don't exactly know the meaning of the feature **"num_compromised"**, so I chose to discard this feature.

The reason that I drop "target" in the below is just to make X a test set from the labeled data set.

```
# remove some feature field if I don't actually understand what does it mean (e.g., num_compromised)
# remove 'target' since we're going to create the training set
# X: The traing set (i.e., x_train)
X = df.drop(columns=['num_compromised', 'target'])
```

6. Feature Engineering skills I used

As I mentioned previously, first, I used the parameter **"error_bad_lines"** to deal with missing value when reading data from a txt/csv file.

Also, I used the function **"drop_duplicates()"** to remove the duplicate entries.

Finally, I used LabelEncoder to do label encoding to convert categorical feature to numerical values.

7. ML algorithms I used

- KNN:

  I used KNeignborsClassifier() to build an KNN model to fit the data and did the prediction.

```
In [8]: # Build an K-NN model
        from sklearn.neighbors import KNeighborsClassifier
        knn = KNeighborsClassifier(n_neighbors=5)
        knn.fit(X, y)

Out[8]: KNeighborsClassifier()
```

```
In [14]: # Making prediction
         prediction = knn.predict(test_set)
         print("Predicted label:", prediction)
         print(len(prediction))
         print("Test set score: {:.2f}".format(knn.score(test_set, correct_data)))

         Predicted label: [0 0 0 ... 0 0 0]
         311029
         Test set score: 0.92
```

- SVM:

  I used **LinearSVC** module to build the SVM model since it says it's for classification.

```
In [15]: # Build SVM model
         from sklearn.svm import LinearSVC
         svc = LinearSVC()
         svc.fit(X, y)
         print(svc.coef_)
```

I guess it may be I didn't specify parameters when building the model, it seems like the accuracy of this model is not ideal enough. It may need some further tuning for parameters.

```
In [16]: svc.predict(test_set)
         print("Test set score: {:.2f}".format(svc.score(test_set, correct_data)))

         Test set score: 0.78
```

- Random Forest:

  I tried to build a random forest model with different **"max_depth"** parameter values. The first time I specified it to "3" and the test score was about 0.28 which is terrible.

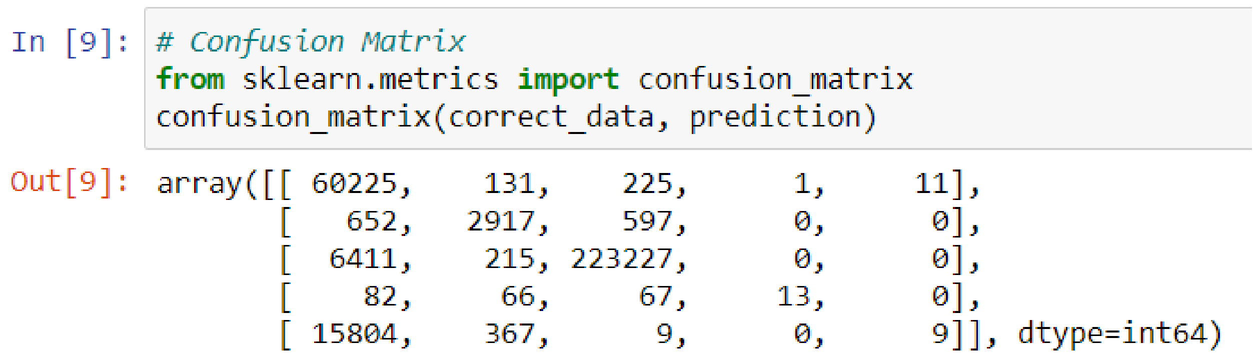
  Finally, when it be set to "10", the score seems more acceptable.

```
In [24]: # Build Random Forest model
         from sklearn.ensemble import RandomForestClassifier
         rfc = RandomForestClassifier(max_depth=10, random_state=0)
         rfc.fit(X, y)

Out[24]: RandomForestClassifier(max_depth=10, random_state=0)

In [25]: rfc.predict(test_set)
         print("Test set score: {:.2f}".format(rfc.score(test_set, correct_data)))

         Test set score: 0.92
```

- By the way, I also tried to use Naïve Bayes, but I then gave up since I encountered MemoryError issue during building the model.

8. Confusion Matrix

The below shows the confusion matrix of (correct_data, prediction):



```
In [9]: # Confusion Matrix
        from sklearn.metrics import confusion_matrix
        confusion_matrix(correct_data, prediction)

Out[9]: array([[ 60225,    131,    225,      1,     11],
               [   652,   2917,    597,      0,      0],
               [  6411,    215, 223227,      0,      0],
               [    82,     66,     67,     13,      0],
               [ 15804,    367,      9,      0,      9]], dtype=int64)
```

9. Discussions and Conclusion

To be honest, this is an interesting lab to do and explore, it a good chance to let students to know how machine learning works and there're many useful built-in algorithms, models available in the platform.

I think the biggest problem I firstly encountered was about data preprocessing, since there're many format issue you need to take care. It makes the first step the most difficult.

Also, I think the lab may still be a little frustrated for some people who never manipulate machine learning before picking this course at the first place. I would recommend this course to add one or two little labs or tutorials to teach students how to manipulate the basic functions in Sklearn and the basic steps to build, train, and evaluate the model before doing this lab1. Otherwise, they may need to spend lots of times to get familiar with those APIs.

Overall, I still enjoyed this lab especially after seeing the prediction result of the model.