Improve Accuracy

There are 2 ways to improve the accuracy of machine learning model

Choose the right algorithm

Different algorithms have different strengths and weaknesses, and they may perform better or worse depending on the data size, distribution, complexity, and noise.

- Decision trees are simple and intuitive, but they can be prone to overfitting and instability.
- Logistic regression is fast and easy to interpret, but it can be limited by linearity and multicollinearity.
- KNN is simple and powerful, requires no traning time, but memory intensive, slow.
- Support vector machines are powerful and flexible, but they can be computationally expensive and sensitive to parameter tuning.

For example, in question 1, we build KNN classification model for Titanic (version 1) dataset. However, the average accuraccy is 0.72. We can use another model to increase this number such as decision tree.

```
# DecisionTree
model = DecisionTreeClassifier()
model.fit(X_train,y_train)

y_predict = model.predict(X_test)

# evaluation
print(classification_report(y_test,y_predict))
```

```
precision    recall  f1-score   support

         0.0       0.00      0.00      0.00         0
         1.0       0.71      0.58      0.64        43
         2.0       0.76      0.94      0.84        17
         3.0       0.84      0.87      0.85       119

    accuracy                           0.80       179
   macro avg       0.58      0.60      0.58       179
weighted avg       0.81      0.80      0.80       179
```

As the result, the accuracy reach 0.80

Using one-hot encoding instead of label encoding

When we process Titanic dataset (train.csv), we use label endcoding to normalize data. However, The drawback of Label Encoding is that it assigns labels to numerical values, and the model may misunderstand that there's a relationship between labels based on their numerical values, leading to inaccurate predictions.

To increase the accuracy, we can use one-hot encoding

```
# label the object data
from sklearn.preprocessing import OneHotEncoder

# Object data
object_cols = df.select_dtypes(include=['object']).columns

# init OneHotEncoder
encoder = OneHotEncoder()

# encode the object data
encoded_data = encoder.fit_transform(df[object_cols])

# dataframe of encoded
encoded_df = pd.DataFrame(encoded_data.toarray(),
columns=encoder.get_feature_names_out(object_cols))

# concat encoded_df and df
df = pd.concat([df.select_dtypes(exclude=['object']), encoded_df], axis=1)
```

The accuracy will be like this:

```
precision    recall  f1-score   support

         0.0       1.00      1.00      1.00        60
         1.0       1.00      1.00      1.00       119

    accuracy                           1.00       179
   macro avg       1.00      1.00      1.00       179
weighted avg       1.00      1.00      1.00       179
```

The output looks great ! However, this only occurs when the dataset is simple and less of data point (the Titanic has only more than 800 rows). The larger size of dataset, the more significantly accuracy will fluctuate.