Project Deliverable 2 Code

For ease of use, I uploaded the csv to github, so the code can be ran easily without downloading the dataset or inserting kaggle api key.

Data Preprocessing and Exploration

This section removes any invalid and duplicate entries in our dataset, as well as removing A_id, as it will not be useful for our model. (It is only an identifier.)

In [2]:
```
import pandas as pd
# load the dataset
url = 'https://raw.githubusercontent.com/johnxminimo/cs577applequalityproject/main/apple_quality.csv'

dataset = pd.read_csv(url)
dataset.head()
dataset.describe()
```

Out[2]:

	A_id	Size	Weight	Sweetness	Crunchiness	Juiciness	Ripeness
count	4000.000000	4000.000000	4000.000000	4000.000000	4000.000000	4000.000000	4000.000000
mean	1999.500000	-0.503015	-0.989547	-0.470479	0.985478	0.512118	0.498277
std	1154.844867	1.928059	1.602507	1.943441	1.402757	1.930286	1.874427
min	0.000000	-7.151703	-7.149848	-6.894485	-6.055058	-5.961897	-5.864599
25%	999.750000	-1.816765	-2.011770	-1.738425	0.062764	-0.801286	-0.771677
50%	1999.500000	-0.513703	-0.984736	-0.504758	0.998249	0.534219	0.503445
75%	2999.250000	0.805526	0.030976	0.801922	1.894234	1.835976	1.766212
max	3999.000000	6.406367	5.790714	6.374916	7.619852	7.364403	7.237837

In [3]:
```
# Let's perform some cleaning/preprocessing (removing duplicates, null/invalid records, and features )
# lets first remove duplicates, from our dataset
                                    need apple_id, as this is just an identifier
dataset.drop_duplicates(inplace=True)
```

```
dataset.drop("A_id", axis=1, inplace=True)
dataset.describe()
```

Out[3]:

	Size	Weight	Sweetness	Crunchiness	Juiciness	Ripeness
count	4000.000000	4000.000000	4000.000000	4000.000000	4000.000000	4000.000000
mean	-0.503015	-0.989547	-0.470479	0.985478	0.512118	0.498277
std	1.928059	1.602507	1.943441	1.402757	1.930286	1.874427
min	-7.151703	-7.149848	-6.894485	-6.055058	-5.961897	-5.864599
25%	-1.816765	-2.011770	-1.738425	0.062764	-0.801286	-0.771677
50%	-0.513703	-0.984736	-0.504758	0.998249	0.534219	0.503445
75%	0.805526	0.030976	0.801922	1.894234	1.835976	1.766212
max	6.406367	5.790714	6.374916	7.619852	7.364403	7.237837

In [4]:
```
# As shown here, our dataset rates our apple as either good or bad.
print(dataset["Quality"])
```

```
0       good
1       good
2        bad
3       good
4       good
        ...
3996    good
3997     bad
3998    good
3999    good
4000     NaN
Name: Quality, Length: 4001, dtype: object
```

In [5]:
```
# Instead we should use 1 for good and 0 for bad
dataset["Quality"].replace(("good", "bad"), [1,0], inplace = True)
print(dataset["Quality"])
```

```
0       1.0
1       1.0
2       0.0
3       1.0
4       1.0
       ...
3996    1.0
3997    0.0
3998    1.0
3999    1.0
4000    NaN
Name: Quality, Length: 4001, dtype: float64
```

In [6]:
```
dataset['Acidity']=pd.to_numeric(dataset.Acidity,errors='coerce')
dataset.dropna(inplace=True)
```

In [7]:
```
from sklearn.model_selection import train_test_split
# now lets begin by splitting our dataset into training and testing
X = dataset.drop("Quality", axis = 1)
y = dataset["Quality"]
dataset.info()
print(X)
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 4000 entries, 0 to 3999
Data columns (total 8 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   Size         4000 non-null   float64
 1   Weight       4000 non-null   float64
 2   Sweetness    4000 non-null   float64
 3   Crunchiness  4000 non-null   float64
 4   Juiciness    4000 non-null   float64
 5   Ripeness     4000 non-null   float64
 6   Acidity      4000 non-null   float64
 7   Quality      4000 non-null   float64
dtypes: float64(8)
memory usage: 281.2 KB
          Size    Weight  Sweetness  Crunchiness  Juiciness  Ripeness  \
0    -3.970049 -2.512336   5.346330    -1.012009   1.844900  0.329840   
1    -1.195217 -2.839257   3.664059     1.588232   0.853286  0.867530   
2    -0.292024 -1.351282  -1.738429    -0.342616   2.838636 -0.038033   
3    -0.657196 -2.271627   1.324874    -0.097875   3.637970 -3.413761   
4     1.364217 -1.296612  -0.384658    -0.553006   3.030874 -1.303849   
...        ...       ...        ...          ...        ...       ...   
3995  0.059386 -1.067408  -3.714549     0.473052   1.697986  2.244055   
3996 -0.293118  1.949253  -0.204020    -0.640196   0.024523 -1.087900   
3997 -2.634515 -2.138247  -2.440461     0.657223   2.199709  4.763859   
3998 -4.008004 -1.779337   2.366397    -0.200329   2.161435  0.214488   
3999  0.278540 -1.715505   0.121217    -1.154075   1.266677 -0.776571   

       Acidity  
0    -0.491590  
1    -0.722809  
2     2.621636  
3     0.790723  
4     0.501984  
...        ...  
3995  0.137784  
3996  1.854235  
3997 -1.334611  
3998 -2.229720  
3999  1.599796  

[4000 rows x 7 columns]
```


Preparing and training our first model (Logistic Reg)

For the first model, I opted to train logistic regression, since it is a simple model and could serve as a "baseline".

The methodolgy I chose is to split the data into training and testing set using a 70/30 split. The reason why I chose this split is becaues we have quite a bit of entries (4000), and should be large enough to where we don't need to add more into our testing set or to use cross validation.

I then split the training set into training and tuning, with 20% of the training set going to tuning.

As for hyperparamters, I will be using gridsearch in order to test: l1: lasso reg l2: ridge reg

regulaization strengths (C): 10^-x for x = -5 to 5 (same parameter set from previous homework)

solvers: liblinear

As for determining whether a model is good, I will use precision since we want to ensure that false positives are at a minimum.

In [19]:
```
from numpy import loadtxt
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import classification_report, accuracy_score, confusion_matrix
from sklearn import metrics
from sklearn.metrics import make_scorer, precision_score


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
paramGridLR = {
    'penalty': ['l1', 'l2'],
    'C': [0.001, 0.01, 0.1, 1, 10, 100],
    'solver': ['liblinear'],
}

precision_scorer = make_scorer(precision_score, pos_label=1)
logRes = LogisticRegression()
logResGridSearch = GridSearchCV(estimator=logRes, param_grid = paramGridLR, verbose=1, scoring=precision_scor
logResGridSearch.fit(X_train, y_train)

bestLogResModel = logResGridSearch.best_estimator_
precisionOnTest = precision_score(y_test, bestLogResModel.predict(X_test), pos_label=1)

print("Best parameters for logistic regression found by GridSearch" + str(logResGridSearch.best_params_))
```

```
print("Precision for log reg on validation set using best parameters found by gridSearch: " + str(logResGridS
print("Precision score on test set: " + str(precisionOnTest))
print("Accuracy score on test set:" + str(bestLogResModel.score(X_test, y_test)))
```

```
Fitting 5 folds for each of 12 candidates, totalling 60 fits
/Users/johnminimo/anaconda3/lib/python3.11/site-packages/sklearn/metrics/_classification.py:1469: UndefinedMe
tricWarning: Precision is ill-defined and being set to 0.0 due to no predicted samples. Use `zero_division` p
arameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/Users/johnminimo/anaconda3/lib/python3.11/site-packages/sklearn/metrics/_classification.py:1469: UndefinedMe
tricWarning: Precision is ill-defined and being set to 0.0 due to no predicted samples. Use `zero_division` p
arameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/Users/johnminimo/anaconda3/lib/python3.11/site-packages/sklearn/metrics/_classification.py:1469: UndefinedMe
tricWarning: Precision is ill-defined and being set to 0.0 due to no predicted samples. Use `zero_division` p
arameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/Users/johnminimo/anaconda3/lib/python3.11/site-packages/sklearn/metrics/_classification.py:1469: UndefinedMe
tricWarning: Precision is ill-defined and being set to 0.0 due to no predicted samples. Use `zero_division` p
arameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/Users/johnminimo/anaconda3/lib/python3.11/site-packages/sklearn/metrics/_classification.py:1469: UndefinedMe
tricWarning: Precision is ill-defined and being set to 0.0 due to no predicted samples. Use `zero_division` p
arameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
Best parameters for logistic regression found by GridSearch{'C': 0.01, 'penalty': 'l1', 'solver': 'liblinea
r'}
Precision for log reg on validation set using best parameters found by gridSearch: 0.7673342533707068
Precision score on test set: 0.7689393939393939
Accuracy score on test set:0.7308333333333333
```

LogReg Results

As for the logistic regression results, it seems that the best parameters are: C = 0.01, penalty: l1 (lasso reg).

Test Set

Precision score: 0.768 Accuracy: 0.73

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

Training Second Model Random Forest

For the second model to test, I opted to use Random Forest.

I also chose to use gridsearch inorder to test different hyper parameters like in our logistic regression model.

In [44]:
```
from sklearn.ensemble import RandomForestClassifier

randForestParam = {
    'n_estimators': [100, 200, 500],
    'max_features': ['auto', 'log2'],
    'max_depth': [None, 10, 20],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'bootstrap': [True, False],
    'criterion': ['mse', 'log_loss']
}

randomForestGrid = GridSearchCV(estimator=RandomForestClassifier(), param_grid=randForestParam, cv=5, n_jobs
randomForestGrid.fit(X_train, y_train)

bestRFModel = randomForestGrid.best_estimator_
rfPrecisionOnTest = precision_score(y_test, randomForestGrid.predict(X_test), pos_label=1)


print("Best parameters for random forest found by GridSearch" + str(randomForestGrid.best_params_))
print("Precision for random forest on validation set using best parameters found by gridSearch: " + str(rando
print("Precision score on test set: " + str(rfPrecisionOnTest))
print("Accuracy score on test set:" + str(bestRFModel.score(X_test, y_test)))
```

```
/Users/johnminimo/anaconda3/lib/python3.11/site-packages/sklearn/model_selection/_validation.py:425: FitFaile
dWarning:
2430 fits failed out of a total of 3240.
The score on these train-test partitions for these parameters will be set to nan.
If these failures are not expected, you can try to debug them by setting error_score='raise'.

Below are more details about the failures:
--------------------------------------------------------------------------------
974 fits failed with the following error:
Traceback (most recent call last):
  File "/Users/johnminimo/anaconda3/lib/python3.11/site-packages/sklearn/model_selection/_validation.py", lin
e 732, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "/Users/johnminimo/anaconda3/lib/python3.11/site-packages/sklearn/base.py", line 1144, in wrapper
    estimator._validate_params()
  File "/Users/johnminimo/anaconda3/lib/python3.11/site-packages/sklearn/base.py", line 637, in _validate_par
ams
    validate_parameter_constraints(
  File "/Users/johnminimo/anaconda3/lib/python3.11/site-packages/sklearn/utils/_param_validation.py", line 9
5, in validate_parameter_constraints
    raise InvalidParameterError(
sklearn.utils._param_validation.InvalidParameterError: The 'criterion' parameter of RandomForestClassifier mu
st be a str among {'gini', 'log_loss', 'entropy'}. Got 'mse' instead.

--------------------------------------------------------------------------------
245 fits failed with the following error:
Traceback (most recent call last):
  File "/Users/johnminimo/anaconda3/lib/python3.11/site-packages/sklearn/model_selection/_validation.py", lin
e 732, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "/Users/johnminimo/anaconda3/lib/python3.11/site-packages/sklearn/base.py", line 1144, in wrapper
    estimator._validate_params()
  File "/Users/johnminimo/anaconda3/lib/python3.11/site-packages/sklearn/base.py", line 637, in _validate_par
ams
    validate_parameter_constraints(
  File "/Users/johnminimo/anaconda3/lib/python3.11/site-packages/sklearn/utils/_param_validation.py", line 9
5, in validate_parameter_constraints
    raise InvalidParameterError(
sklearn.utils._param_validation.InvalidParameterError: The 'criterion' parameter of RandomForestClassifier mu
st be a str among {'log_loss', 'entropy', 'gini'}. Got 'mse' instead.

--------------------------------------------------------------------------------
139 fits failed with the following error:
```



```
  File "/Users/johnminimo/anaconda3/lib/python3.11/site-packages/sklearn/model_selection/_validation.py", lin
```

```
e 732, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "/Users/johnminimo/anaconda3/lib/python3.11/site-packages/sklearn/base.py", line 1144, in wrapper
    estimator._validate_params()
  File "/Users/johnminimo/anaconda3/lib/python3.11/site-packages/sklearn/base.py", line 637, in _validate_par
ams
    validate_parameter_constraints(
  File "/Users/johnminimo/anaconda3/lib/python3.11/site-packages/sklearn/utils/_param_validation.py", line 9
5, in validate_parameter_constraints
    raise InvalidParameterError(
sklearn.utils._param_validation.InvalidParameterError: The 'criterion' parameter of RandomForestClassifier mu
st be a str among {'entropy', 'gini', 'log_loss'}. Got 'mse' instead.

--------------------------------------------------------------------------------
24 fits failed with the following error:
Traceback (most recent call last):
  File "/Users/johnminimo/anaconda3/lib/python3.11/site-packages/sklearn/model_selection/_validation.py", lin
e 732, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "/Users/johnminimo/anaconda3/lib/python3.11/site-packages/sklearn/base.py", line 1144, in wrapper
    estimator._validate_params()
  File "/Users/johnminimo/anaconda3/lib/python3.11/site-packages/sklearn/base.py", line 637, in _validate_par
ams
    validate_parameter_constraints(
  File "/Users/johnminimo/anaconda3/lib/python3.11/site-packages/sklearn/utils/_param_validation.py", line 9
5, in validate_parameter_constraints
    raise InvalidParameterError(
sklearn.utils._param_validation.InvalidParameterError: The 'criterion' parameter of RandomForestClassifier mu
st be a str among {'gini', 'entropy', 'log_loss'}. Got 'mse' instead.

--------------------------------------------------------------------------------
177 fits failed with the following error:
Traceback (most recent call last):
  File "/Users/johnminimo/anaconda3/lib/python3.11/site-packages/sklearn/model_selection/_validation.py", lin
e 732, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "/Users/johnminimo/anaconda3/lib/python3.11/site-packages/sklearn/base.py", line 1144, in wrapper
    estimator._validate_params()
  File "/Users/johnminimo/anaconda3/lib/python3.11/site-packages/sklearn/base.py", line 637, in _validate_par
ams
    validate_parameter_constraints(
  File "/Users/johnminimo/anaconda3/lib/python3.11/site-packages/sklearn/utils/_param_validation.py", line 9
5, in validate_parameter_constraints
    raise InvalidParameterError(
sklearn.utils._param_validation.InvalidParameterError: The 'criterion' parameter of RandomForestClassifier mu
```

```
st be a str among {'entropy', 'log_loss', 'gini'}. Got 'mse' instead.

--------------------------------------------------------------------------------
61 fits failed with the following error:
Traceback (most recent call last):
  File "/Users/johnminimo/anaconda3/lib/python3.11/site-packages/sklearn/model_selection/_validation.py", lin
e 732, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "/Users/johnminimo/anaconda3/lib/python3.11/site-packages/sklearn/base.py", line 1144, in wrapper
    estimator._validate_params()
  File "/Users/johnminimo/anaconda3/lib/python3.11/site-packages/sklearn/base.py", line 637, in _validate_par
ams
    validate_parameter_constraints(
  File "/Users/johnminimo/anaconda3/lib/python3.11/site-packages/sklearn/utils/_param_validation.py", line 9
5, in validate_parameter_constraints
    raise InvalidParameterError(
sklearn.utils._param_validation.InvalidParameterError: The 'criterion' parameter of RandomForestClassifier mu
st be a str among {'log_loss', 'gini', 'entropy'}. Got 'mse' instead.

--------------------------------------------------------------------------------
583 fits failed with the following error:
Traceback (most recent call last):
  File "/Users/johnminimo/anaconda3/lib/python3.11/site-packages/sklearn/model_selection/_validation.py", lin
e 732, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "/Users/johnminimo/anaconda3/lib/python3.11/site-packages/sklearn/base.py", line 1144, in wrapper
    estimator._validate_params()
  File "/Users/johnminimo/anaconda3/lib/python3.11/site-packages/sklearn/base.py", line 637, in _validate_par
ams
    validate_parameter_constraints(
  File "/Users/johnminimo/anaconda3/lib/python3.11/site-packages/sklearn/utils/_param_validation.py", line 9
5, in validate_parameter_constraints
    raise InvalidParameterError(
sklearn.utils._param_validation.InvalidParameterError: The 'max_features' parameter of RandomForestClassifier
must be an int in the range [1, inf), a float in the range (0.0, 1.0], a str among {'sqrt', 'log2'} or None.
Got 'auto' instead.

--------------------------------------------------------------------------------
227 fits failed with the following error:
Traceback (most recent call last):
  File "/Users/johnminimo/anaconda3/lib/python3.11/site-packages/sklearn/model_selection/_validation.py", lin
e 732, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "/Users/johnminimo/anaconda3/lib/python3.11/site-packages/sklearn/base.py", line 1144, in wrapper
    estimator._validate_params()
```

```
  File "/Users/johnminimo/anaconda3/lib/python3.11/site-packages/sklearn/base.py", line 637, in _validate_par
ams
    validate_parameter_constraints(
  File "/Users/johnminimo/anaconda3/lib/python3.11/site-packages/sklearn/utils/_param_validation.py", line 9
5, in validate_parameter_constraints
    raise InvalidParameterError(
sklearn.utils._param_validation.InvalidParameterError: The 'max_features' parameter of RandomForestClassifier
must be an int in the range [1, inf), a float in the range (0.0, 1.0], a str among {'log2', 'sqrt'} or None.
Got 'auto' instead.

  warnings.warn(some_fits_failed_message, FitFailedWarning)
/Users/johnminimo/anaconda3/lib/python3.11/site-packages/sklearn/model_selection/_search.py:976: UserWarning:
One or more of the test scores are non-finite: [       nan        nan        nan        nan        nan
nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan 0.87178571 0.875      0.87107143
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

```
 0.87321429 0.87214286 0.87321429 0.86678571 0.86571429 0.86964286
 0.8725     0.86964286 0.87142857 0.86928571 0.87285714 0.87
 0.86464286 0.86785714 0.87       0.86642857 0.86678571 0.86642857
 0.86464286 0.86642857 0.87035714 0.86428571 0.86714286 0.86464286
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan 0.86464286 0.86642857 0.86464286
 0.86178571 0.865      0.86642857 0.86642857 0.8625     0.86392857
 0.86285714 0.86821429 0.86392857 0.86357143 0.86285714 0.86535714
 0.86035714 0.86428571 0.86321429 0.86607143 0.86428571 0.86214286
 0.86107143 0.86428571 0.86428571 0.86392857 0.85892857 0.86285714
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan 0.87107143 0.8725     0.87035714
 0.87392857 0.87214286 0.87107143 0.87392857 0.86714286 0.86928571
 0.86964286 0.86892857 0.87071429 0.86607143 0.86964286 0.87214286
 0.86964286 0.86892857 0.87214286 0.86857143 0.86857143 0.86821429
 0.86785714 0.8675     0.86785714 0.86464286 0.86857143 0.86678571
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
```

```
       nan        nan        nan        nan        nan        nan
       nan        nan        nan        nan        nan        nan
       nan        nan        nan        nan        nan        nan
       nan        nan        nan        nan        nan        nan
       nan        nan        nan        nan        nan        nan
       nan        nan        nan        nan        nan        nan
       nan        nan        nan        nan        nan        nan
       nan        nan        nan        nan        nan        nan
       nan        nan        nan 0.86571429 0.86964286 0.86785714
 0.865      0.86571429 0.86428571 0.86285714 0.86392857 0.86607143
 0.86714286 0.85928571 0.86464286 0.85964286 0.86357143 0.86357143
 0.865      0.86107143 0.86428571 0.86392857 0.86214286 0.86285714
 0.86285714 0.86392857 0.86321429 0.85928571 0.85928571 0.86392857
       nan        nan        nan        nan        nan        nan
       nan        nan        nan        nan        nan        nan
       nan        nan        nan        nan        nan        nan
       nan        nan        nan        nan        nan        nan
       nan        nan        nan 0.85964286 0.85892857 0.85785714
 0.85714286 0.85928571 0.85785714 0.85642857 0.85714286 0.85821429
 0.85392857 0.85821429 0.85785714 0.85892857 0.8575     0.85642857
 0.8575     0.85535714 0.85964286 0.85642857 0.85714286 0.8575
 0.85714286 0.85714286 0.85428571 0.85642857 0.85857143 0.855
       nan        nan        nan        nan        nan        nan
       nan        nan        nan        nan        nan        nan
       nan        nan        nan        nan        nan        nan
       nan        nan        nan        nan        nan        nan
       nan        nan        nan 0.86107143 0.86535714 0.86285714
 0.865      0.86178571 0.86428571 0.86607143 0.86214286 0.86392857
 0.86535714 0.85857143 0.86607143 0.86214286 0.8625     0.86535714
 0.86214286 0.86107143 0.8625     0.86107143 0.86714286 0.86071429
 0.86214286 0.86       0.86464286 0.86428571 0.86357143 0.8625    ]
  warnings.warn(
Best parameters for random forest found by GridSearch{'bootstrap': True, 'criterion': 'log_loss', 'max_dept
h': None, 'max_features': 'log2', 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 200}
Precision for random forest on validation set using best parameters found by gridSearch: 0.875
Precision score on test set: 0.8887070376432079
Accuracy score on test set:0.89
```

Results on Random Forest

For random forest results, it seems that the best parameters are:{'bootstrap': True, 'criterion': 'log_loss', 'max_depth': 20, 'max_features': 'log2', 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 500}.

Test Set

Precision score: 0.89 Accuracy: 0.89

Our precision score and accuracy score both do better compared to our first baseline model.

With a precision score of 0.76, we have reduced the amount of false positives substantially, from our initial score of 0.73 with logistic regression.

Our accuracy also increased, which means that we are getting a larger amount of our test set correct, and not just getting less false positives. Our model performs better as a whole.

Training Third Model, ANN

Since ANN is computationally extensive, I am opting not to implement gridsearch, since it will increase the complexity of our code since we need to create a separaate model builder function to work with gridsearch, and also would take long to train + test due to the different parameters and combinations gridsearch would use.

Instead, I am opting to use relu for hidden layers, and then for our final output layer, a sigmoid unit. This is similar to the approach we took in our homework.

In [46]:
```
dataset.dtypes
```

Out[46]:
```
A_id           float64
Size           float64
Weight         float64
Sweetness      float64
Crunchiness    float64
Juiciness      float64
Ripeness       float64
Acidity        float64
Quality        float64
dtype: object
```

In [20]:
```
from keras.losses import BinaryCrossentropy
from keras.optimizers import Adam
from keras import backend
from keras.layers import Dense
```

```
model = Sequential()
model.add(Dense(9, activation='relu', input_shape=(7,)))
model.add(Dense(15, activation='relu'))
model.add(Dense(1, activation='sigmoid'))

model.summary()

model.compile(optimizer=Adam(), loss=BinaryCrossentropy(), metrics=['accuracy'])
model.fit(X_train, y_train, epochs=100)
```

```
Model: "sequential_7"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_27 (Dense)            (None, 9)                 72

 dense_28 (Dense)            (None, 15)                150

 dense_29 (Dense)            (None, 1)                 16

=================================================================
Total params: 238
Trainable params: 238
Non-trainable params: 0
_________________________________________________________________
Epoch 1/100
 1/88 [..............................] - ETA: 23s - loss: 0.6898 - accuracy: 0.6562
2024-04-21 20:31:10.846940: I tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc:114] Plu
gin optimizer for device_type GPU is enabled.
```

```
88/88 [==============================] - 1s 8ms/step - loss: 0.6212 - accuracy: 0.6746
Epoch 2/100
88/88 [==============================] - 1s 7ms/step - loss: 0.5315 - accuracy: 0.7346
Epoch 3/100
88/88 [==============================] - 1s 6ms/step - loss: 0.5237 - accuracy: 0.7311
Epoch 4/100
88/88 [==============================] - 1s 7ms/step - loss: 0.5203 - accuracy: 0.7404
Epoch 5/100
88/88 [==============================] - 1s 6ms/step - loss: 0.5194 - accuracy: 0.7404
Epoch 6/100
88/88 [==============================] - 1s 6ms/step - loss: 0.5181 - accuracy: 0.7379
Epoch 7/100
88/88 [==============================] - 1s 7ms/step - loss: 0.5185 - accuracy: 0.7418
Epoch 8/100
88/88 [==============================] - 1s 6ms/step - loss: 0.5193 - accuracy: 0.7393
Epoch 9/100
88/88 [==============================] - 1s 7ms/step - loss: 0.5190 - accuracy: 0.7389
Epoch 10/100
88/88 [==============================] - 1s 6ms/step - loss: 0.5191 - accuracy: 0.7400
Epoch 11/100
88/88 [==============================] - 1s 7ms/step - loss: 0.5177 - accuracy: 0.7421
Epoch 12/100
88/88 [==============================] - 1s 7ms/step - loss: 0.5210 - accuracy: 0.7396
Epoch 13/100
88/88 [==============================] - 1s 7ms/step - loss: 0.5186 - accuracy: 0.7389
Epoch 14/100
88/88 [==============================] - 1s 7ms/step - loss: 0.5230 - accuracy: 0.7439
Epoch 15/100
88/88 [==============================] - 1s 7ms/step - loss: 0.5202 - accuracy: 0.7411
Epoch 16/100
88/88 [==============================] - 1s 7ms/step - loss: 0.5285 - accuracy: 0.7375
Epoch 17/100
88/88 [==============================] - 1s 7ms/step - loss: 0.5247 - accuracy: 0.7350
Epoch 18/100
88/88 [==============================] - 1s 7ms/step - loss: 0.5273 - accuracy: 0.7314
Epoch 19/100
88/88 [==============================] - 1s 8ms/step - loss: 0.5238 - accuracy: 0.7425
Epoch 20/100
88/88 [==============================] - 1s 7ms/step - loss: 0.5295 - accuracy: 0.7382
Epoch 21/100
88/88 [==============================] - 1s 7ms/step - loss: 0.5263 - accuracy: 0.7332
Epoch 22/100
88/88 [==============================] - 1s 7ms/step - loss: 0.5282 - accuracy: 0.7393
Epoch 23/100
88/88 [==============================] - 1s 7ms/step - loss: 0.5261 - accuracy: 0.7393
```

```
Epoch 24/100
88/88 [==============================] - 1s 7ms/step - loss: 0.5391 - accuracy: 0.7254
Epoch 25/100
88/88 [==============================] - 1s 7ms/step - loss: 0.5562 - accuracy: 0.7229
Epoch 26/100
88/88 [==============================] - 1s 7ms/step - loss: 0.5377 - accuracy: 0.7368
Epoch 27/100
88/88 [==============================] - 1s 7ms/step - loss: 0.5318 - accuracy: 0.7446
Epoch 28/100
88/88 [==============================] - 1s 7ms/step - loss: 0.5272 - accuracy: 0.7371
Epoch 29/100
88/88 [==============================] - 1s 7ms/step - loss: 0.5424 - accuracy: 0.7286
Epoch 30/100
88/88 [==============================] - 1s 7ms/step - loss: 0.5348 - accuracy: 0.7314
Epoch 31/100
88/88 [==============================] - 1s 7ms/step - loss: 0.5279 - accuracy: 0.7393
Epoch 32/100
88/88 [==============================] - 1s 7ms/step - loss: 0.5421 - accuracy: 0.7261
Epoch 33/100
88/88 [==============================] - 1s 8ms/step - loss: 0.5389 - accuracy: 0.7425
Epoch 34/100
88/88 [==============================] - 1s 7ms/step - loss: 0.5576 - accuracy: 0.7246
Epoch 35/100
88/88 [==============================] - 1s 7ms/step - loss: 0.5515 - accuracy: 0.7239
Epoch 36/100
88/88 [==============================] - 1s 7ms/step - loss: 0.5701 - accuracy: 0.7154
Epoch 37/100
88/88 [==============================] - 1s 7ms/step - loss: 0.5543 - accuracy: 0.7329
Epoch 38/100
88/88 [==============================] - 1s 7ms/step - loss: 0.5437 - accuracy: 0.7289
Epoch 39/100
88/88 [==============================] - 1s 7ms/step - loss: 0.5535 - accuracy: 0.7211
Epoch 40/100
88/88 [==============================] - 1s 7ms/step - loss: 0.5487 - accuracy: 0.7271
Epoch 41/100
88/88 [==============================] - 1s 8ms/step - loss: 0.5542 - accuracy: 0.7261
Epoch 42/100
88/88 [==============================] - 1s 8ms/step - loss: 0.5978 - accuracy: 0.7032
Epoch 43/100
88/88 [==============================] - 1s 8ms/step - loss: 0.5398 - accuracy: 0.7332
Epoch 44/100
88/88 [==============================] - 1s 8ms/step - loss: 0.5554 - accuracy: 0.7211
Epoch 45/100
88/88 [==============================] - 1s 7ms/step - loss: 0.5614 - accuracy: 0.7236
```

```
88/88 [==============================] - 1s 7ms/step - loss: 0.5502 - accuracy: 0.7246
Epoch 47/100
88/88 [==============================] - 1s 7ms/step - loss: 0.5738 - accuracy: 0.7157
Epoch 48/100
88/88 [==============================] - 1s 7ms/step - loss: 0.5463 - accuracy: 0.7275
Epoch 49/100
88/88 [==============================] - 1s 7ms/step - loss: 0.5472 - accuracy: 0.7293
Epoch 50/100
88/88 [==============================] - 1s 7ms/step - loss: 0.6441 - accuracy: 0.6929
Epoch 51/100
88/88 [==============================] - 1s 7ms/step - loss: 0.5655 - accuracy: 0.7164
Epoch 52/100
88/88 [==============================] - 1s 7ms/step - loss: 0.5633 - accuracy: 0.7200
Epoch 53/100
88/88 [==============================] - 1s 7ms/step - loss: 0.5804 - accuracy: 0.7136
Epoch 54/100
88/88 [==============================] - 1s 7ms/step - loss: 0.5618 - accuracy: 0.7146
Epoch 55/100
88/88 [==============================] - 1s 7ms/step - loss: 0.6081 - accuracy: 0.7014
Epoch 56/100
88/88 [==============================] - 1s 8ms/step - loss: 0.5742 - accuracy: 0.7161
Epoch 57/100
88/88 [==============================] - 1s 7ms/step - loss: 0.5409 - accuracy: 0.7271
Epoch 58/100
88/88 [==============================] - 1s 7ms/step - loss: 0.6361 - accuracy: 0.7050
Epoch 59/100
88/88 [==============================] - 1s 9ms/step - loss: 0.6375 - accuracy: 0.7029
Epoch 60/100
88/88 [==============================] - 1s 9ms/step - loss: 0.5986 - accuracy: 0.7018
Epoch 61/100
88/88 [==============================] - 1s 8ms/step - loss: 0.5648 - accuracy: 0.7250
Epoch 62/100
88/88 [==============================] - 1s 7ms/step - loss: 0.5504 - accuracy: 0.7289
Epoch 63/100
88/88 [==============================] - 1s 7ms/step - loss: 0.5596 - accuracy: 0.7246
Epoch 64/100
88/88 [==============================] - 1s 7ms/step - loss: 0.5775 - accuracy: 0.7146
Epoch 65/100
88/88 [==============================] - 1s 7ms/step - loss: 0.5597 - accuracy: 0.7211
Epoch 66/100
88/88 [==============================] - 1s 7ms/step - loss: 0.6129 - accuracy: 0.7018
Epoch 67/100
88/88 [==============================] - 1s 7ms/step - loss: 0.5659 - accuracy: 0.7214
Epoch 68/100
88/88 [==============================] - 1s 7ms/step - loss: 0.5511 - accuracy: 0.7257
```

```
Epoch 69/100
88/88 [==============================] - 1s 7ms/step - loss: 0.6584 - accuracy: 0.6871
Epoch 70/100
88/88 [==============================] - 1s 7ms/step - loss: 0.5528 - accuracy: 0.7254
Epoch 71/100
88/88 [==============================] - 1s 7ms/step - loss: 0.6306 - accuracy: 0.6964
Epoch 72/100
88/88 [==============================] - 1s 7ms/step - loss: 0.6023 - accuracy: 0.7014
Epoch 73/100
88/88 [==============================] - 1s 7ms/step - loss: 0.6219 - accuracy: 0.6993
Epoch 74/100
88/88 [==============================] - 1s 7ms/step - loss: 0.5940 - accuracy: 0.7207
Epoch 75/100
88/88 [==============================] - 1s 7ms/step - loss: 0.6068 - accuracy: 0.7036
Epoch 76/100
88/88 [==============================] - 1s 7ms/step - loss: 0.6657 - accuracy: 0.6843
Epoch 77/100
88/88 [==============================] - 1s 7ms/step - loss: 0.5514 - accuracy: 0.7293
Epoch 78/100
88/88 [==============================] - 1s 7ms/step - loss: 0.5840 - accuracy: 0.7186
Epoch 79/100
88/88 [==============================] - 1s 8ms/step - loss: 0.5918 - accuracy: 0.7196
Epoch 80/100
88/88 [==============================] - 1s 8ms/step - loss: 0.6100 - accuracy: 0.6993
Epoch 81/100
88/88 [==============================] - 1s 7ms/step - loss: 0.5764 - accuracy: 0.7075
Epoch 82/100
88/88 [==============================] - 1s 7ms/step - loss: 0.6168 - accuracy: 0.7050
Epoch 83/100
88/88 [==============================] - 1s 7ms/step - loss: 0.5828 - accuracy: 0.7111
Epoch 84/100
88/88 [==============================] - 1s 7ms/step - loss: 0.6435 - accuracy: 0.6957
Epoch 85/100
88/88 [==============================] - 1s 7ms/step - loss: 0.6247 - accuracy: 0.6979
Epoch 86/100
88/88 [==============================] - 1s 7ms/step - loss: 0.5863 - accuracy: 0.7075
Epoch 87/100
88/88 [==============================] - 1s 7ms/step - loss: 0.6012 - accuracy: 0.7093
Epoch 88/100
88/88 [==============================] - 1s 7ms/step - loss: 0.5602 - accuracy: 0.7161
Epoch 89/100
88/88 [==============================] - 1s 6ms/step - loss: 0.6011 - accuracy: 0.7054
Epoch 90/100
88/88 [==============================] - 1s 6ms/step - loss: 0.5792 - accuracy: 0.7061
```

```
88/88 [==============================] - 1s 7ms/step - loss: 0.6519 - accuracy: 0.6986
Epoch 92/100
88/88 [==============================] - 1s 7ms/step - loss: 0.5918 - accuracy: 0.7054
Epoch 93/100
88/88 [==============================] - 1s 8ms/step - loss: 0.5987 - accuracy: 0.7064
Epoch 94/100
88/88 [==============================] - 1s 7ms/step - loss: 0.6248 - accuracy: 0.6957
Epoch 95/100
88/88 [==============================] - 1s 6ms/step - loss: 0.5759 - accuracy: 0.7211
Epoch 96/100
88/88 [==============================] - 1s 7ms/step - loss: 0.5930 - accuracy: 0.7111
Epoch 97/100
88/88 [==============================] - 1s 7ms/step - loss: 0.6248 - accuracy: 0.6996
Epoch 98/100
88/88 [==============================] - 1s 7ms/step - loss: 0.6379 - accuracy: 0.6882
Epoch 99/100
88/88 [==============================] - 1s 8ms/step - loss: 0.6256 - accuracy: 0.6950
Epoch 100/100
88/88 [==============================] - 1s 7ms/step - loss: 0.6191 - accuracy: 0.7114
```

Out[20]:
```
<keras.callbacks.History at 0x2bf2fab90>
```

In [22]:
```
model.evaluate(X_test, y_test)
yPredict = model.predict(X_test)
```

```
38/38 [==============================] - 0s 4ms/step - loss: 0.5412 - accuracy: 0.7425
38/38 [==============================] - 0s 2ms/step
2024-04-21 20:33:31.116422: I tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc:114] Plu
gin optimizer for device_type GPU is enabled.
```

In [23]:
```
print(yPredict) # if > .5, make = 1
```

```
[[0.5376683 ]
 [0.8730905 ]
 [0.9414666 ]
 ...
 [0.6869172 ]
 [0.45628667]
 [0.5662097 ]]
```

In [28]:
```
finalPredict = [1 if y > 0.5 else 0 for y in yPredict]
#print(finalPredict)

print(classification_report(y_test, finalPredict))
```

	precision	recall	f1-score	support
0.0	0.73	0.76	0.74	593
1.0	0.75	0.73	0.74	607
accuracy			0.74	1200
macro avg	0.74	0.74	0.74	1200
weighted avg	0.74	0.74	0.74	1200

ANN Results on Test

Accuracy of 0.74 Precision of 0.73

Conclusion

The best model for our problem based on the testing completed, seems to be random forests. With a high accuracy of 0.89 and a precision of 0.89, this model best suited our needs.

With such a high precision, we know that we are finding less false positives, which means that bad apples are not able to get through in our predictions, while our accuracy is also great, further proving that the model is not a fluke.