baseline_anlaysis

February 23, 2023

[ ]:

```
import numpy as np
import pandas as pd

from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
import xgboost as xgb
from sklearn.tree import DecisionTreeClassifier
import catboost
import lightgbm
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix,␣
 ↪classification_report

np.random.seed(0)
```

```
/Users/ryanshea/opt/miniconda3/envs/tensorflow/lib/python3.8/site-
packages/xgboost/compat.py:36: FutureWarning: pandas.Int64Index is deprecated
and will be removed from pandas in a future version. Use pandas.Index with the
appropriate dtype instead.
  from pandas import MultiIndex, Int64Index
```

0.1 Notes:

- Generally, most ML models are able to find a difference between the two classes
- Logistic Regression and Decision Tress have lower accuracies but even they do better than random guessing
- None of the models have been tuned but most of the others have around .77 accuracy
- Shows that there is a lot of room for improvement in data synthesis

[ ]:

```
fake = pd.read_csv('fake_returns.csv').drop("Unnamed: 0", axis=1).T
real = pd.read_csv('real_returns.csv').drop("Unnamed: 0", axis=1).T

fake['label'] = 0
real['label'] = 1
print(fake.head())
print(real.head())
```

```
          0         1         2         3         4  label
0  0.018903 -0.028697  0.016171 -0.010939  0.024098      0
1  0.004445  0.002597  0.011681  0.006875  0.002442      0
2  0.004113 -0.000945  0.009640 -0.002228  0.002092      0
3  0.003525 -0.005657 -0.001579 -0.000251  0.006937      0
4 -0.003574 -0.003970 -0.007127 -0.002242  0.001875      0
          0         1         2         3         4  label
0 -0.015061  0.000583  0.005816  0.001912 -0.002144      1
1 -0.067705 -0.020872  0.043075  0.007002 -0.000873      1
2  0.018355  0.010048 -0.011197  0.003540 -0.009258      1
3  0.000220 -0.011784  0.011652  0.013388  0.026251      1
4  0.004243  0.002334 -0.017521  0.017447  0.006675      1
```

[ ]:

```
# combine, shuffle, and split
np.random.seed(0)
df = pd.concat([fake, real])
df = df.sample(frac=1).reset_index(drop=True)
df
```

[ ]:

```
              0         1         2         3         4  label
0     -0.005124 -0.022862 -0.015359 -0.053346  0.008409      1
1     -0.006478 -0.015799  0.022404  0.014828  0.035294      1
2     -0.010542 -0.014620 -0.007076 -0.004644 -0.005648      0
3     -0.000238  0.000353  0.013898  0.004238 -0.002228      0
4     -0.018514  0.004395  0.011364  0.003803  0.000000      1
…           …         …         …         …         …    …
99631 -0.002732 -0.001971 -0.000224 -0.004348 -0.000190      0
99632 -0.007494 -0.009779  0.006263  0.001235  0.014107      0
99633 -0.014397 -0.000361  0.008395 -0.002989 -0.010194      0
99634 -0.003073 -0.005414 -0.000081 -0.005265 -0.004561      0
99635  0.026337  0.001496  0.005961 -0.003383  0.022250      1

[99636 rows x 6 columns]
```

[ ]:

```
print(df['label'].sum() / len(df)) # roughly 50/50 split
```

```
0.4981733509976314
```

[ ]:

```
X_train, X_test, y_train, y_test = train_test_split(df.drop('label', axis=1),␣
  ↪df['label'], test_size=0.1, random_state=0)
```

[ ]:

```
def eval_model(model):
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    print(f"Accuracy: {accuracy_score(y_test, y_pred)}\n\n")
    print(f"Confusion Matrix:\n{confusion_matrix(y_test, y_pred)}\n\n")
    print(f"Classification Report:\n\n{classification_report(y_test, y_pred)}")
    return model
```

[ ]:

```
logistic = eval_model(LogisticRegression())
```

```
Accuracy: 0.5630268968285829


Confusion Matrix:
[[3218 1820]
 [2534 2392]]


Classification Report:

              precision    recall  f1-score   support

           0       0.56      0.64      0.60      5038
           1       0.57      0.49      0.52      4926

    accuracy                           0.56      9964
   macro avg       0.56      0.56      0.56      9964
weighted avg       0.56      0.56      0.56      9964
```

[ ]:

```
svm = eval_model(SVC())
```

```
Accuracy: 0.7760939381774388


Confusion Matrix:
[[3894 1144]
 [1087 3839]]


Classification Report:

              precision    recall  f1-score   support

           0       0.78      0.77      0.78      5038
           1       0.77      0.78      0.77      4926

    accuracy                           0.78      9964
   macro avg       0.78      0.78      0.78      9964
weighted avg       0.78      0.78      0.78      9964
```

[ ]:

```
rf = eval_model(RandomForestClassifier())
```

```
Accuracy: 0.7786029706945002


Confusion Matrix:
[[3695 1343]
 [ 863 4063]]


Classification Report:

              precision    recall  f1-score   support

           0       0.81      0.73      0.77      5038
           1       0.75      0.82      0.79      4926

    accuracy                           0.78      9964
   macro avg       0.78      0.78      0.78      9964
weighted avg       0.78      0.78      0.78      9964
```

```
[ ]: xgb = eval_model(xgb.XGBClassifier())
```

```
/Users/ryanshea/opt/miniconda3/envs/tensorflow/lib/python3.8/site-
packages/xgboost/sklearn.py:1224: UserWarning: The use of label encoder in
XGBClassifier is deprecated and will be removed in a future release. To remove
this warning, do the following: 1) Pass option use_label_encoder=False when
constructing XGBClassifier object; and 2) Encode your labels (y) as integers
starting with 0, i.e. 0, 1, 2, …, [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)
/Users/ryanshea/opt/miniconda3/envs/tensorflow/lib/python3.8/site-
packages/xgboost/data.py:250: FutureWarning: pandas.Int64Index is deprecated and
will be removed from pandas in a future version. Use pandas.Index with the
appropriate dtype instead.
  elif isinstance(data.columns, (pd.Int64Index, pd.RangeIndex)):
```

```
[00:31:45] WARNING: ../src/learner.cc:1115: Starting in XGBoost 1.3.0, the
default evaluation metric used with the objective 'binary:logistic' was changed
from 'error' to 'logloss'. Explicitly set eval_metric if you'd like to restore
the old behavior.
Accuracy: 0.7768968285828984


Confusion Matrix:
[[3784 1254]
 [ 969 3957]]


Classification Report:
```

```
              precision    recall  f1-score   support

           0       0.80      0.75      0.77      5038
           1       0.76      0.80      0.78      4926

    accuracy                           0.78      9964
   macro avg       0.78      0.78      0.78      9964
weighted avg       0.78      0.78      0.78      9964
```

```
[ ]: tree = eval_model(DecisionTreeClassifier())
```

```
Accuracy: 0.6964070654355681


Confusion Matrix:
[[3457 1581]
 [1444 3482]]


Classification Report:

              precision    recall  f1-score   support

           0       0.71      0.69      0.70      5038
           1       0.69      0.71      0.70      4926

    accuracy                           0.70      9964
   macro avg       0.70      0.70      0.70      9964
weighted avg       0.70      0.70      0.70      9964
```

```
[ ]: cat = eval_model(catboost.CatBoostClassifier(verbose=False))
```

```
Accuracy: 0.7785026093938178


Confusion Matrix:
[[3761 1277]
 [ 930 3996]]


Classification Report:

              precision    recall  f1-score   support

           0       0.80      0.75      0.77      5038
           1       0.76      0.81      0.78      4926

    accuracy                           0.78      9964
   macro avg       0.78      0.78      0.78      9964
weighted avg       0.78      0.78      0.78      9964
```

[ ]:

```python
light = eval_model(lightgbm.LGBMClassifier())
```

```
Accuracy: 0.779205138498595


Confusion Matrix:
[[3724 1314]
 [ 886 4040]]


Classification Report:

              precision    recall  f1-score   support

           0       0.81      0.74      0.77      5038
           1       0.75      0.82      0.79      4926

    accuracy                           0.78      9964
   macro avg       0.78      0.78      0.78      9964
weighted avg       0.78      0.78      0.78      9964
```