Brian Elinsky

Professor Lawrence Fulton, Ph.D.

2020SP_MSDS_422-DL_SEC55

5 May 2020

Assignment 3: Evaluating Classification Models

PROBLEM DESCRIPTION

Given an array of pixels and their associated degree of darkness, the objective is to classify each image as the correct digit. As a manager, I need to decide if building a PCA model is worth the added time, or if my team should spend more time on the predictive model.

RESEARCH DESIGN AND MODELING METHODS

The experiment had us first fitting a random forest classifier. Second, train a PCA on both the training and testing data. Third, use the output of the PCA as features to train a new random forest classifier. Finally, the objective was to correct the flaw in the original experiment design.

DATA PREPARATION

I did very little pre-processing. The first random forest classifier had no pre-processing steps. I utilized a StandardScaler for the PCA.

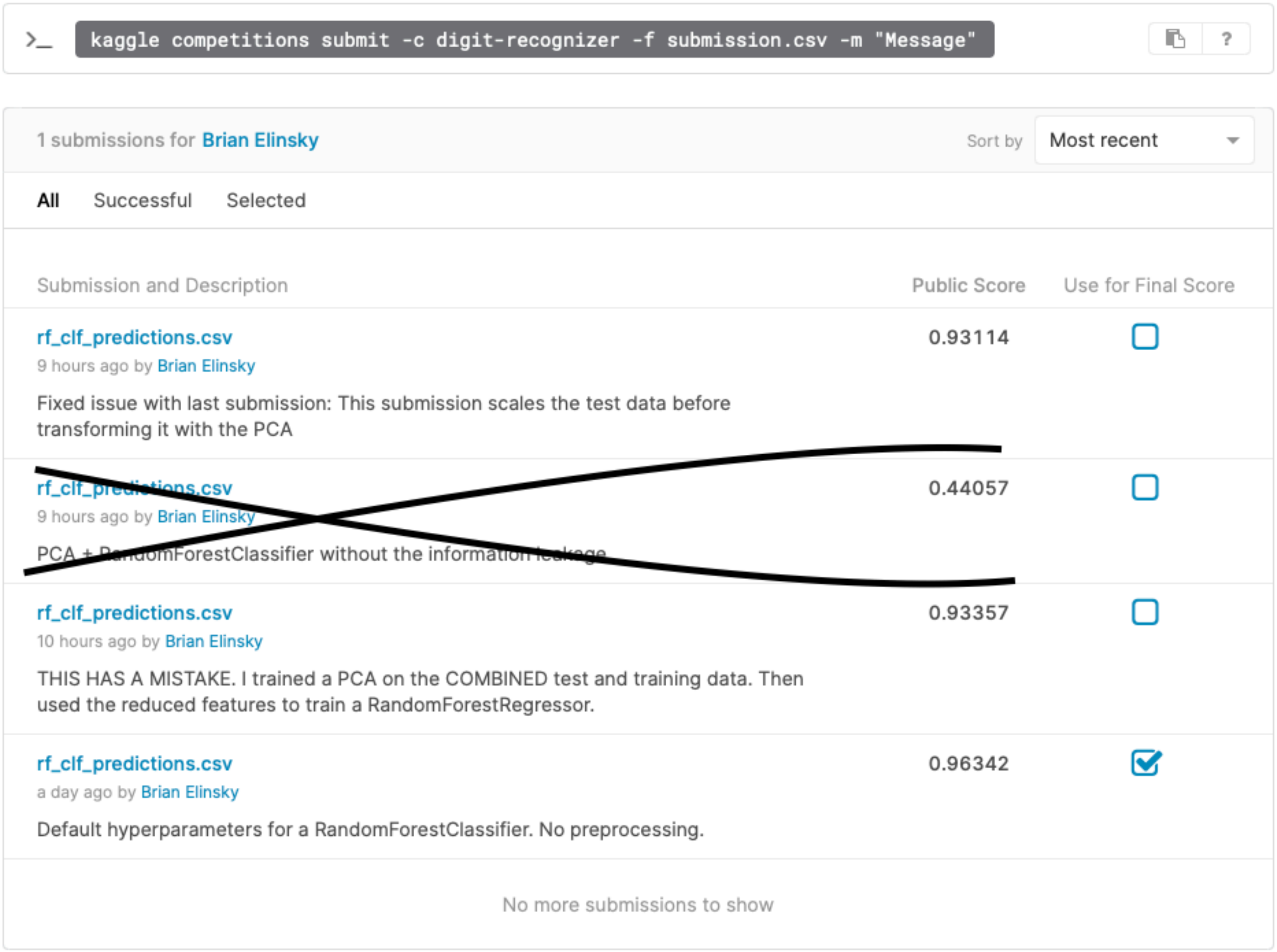
RESULTS AND MODEL EVALUATION

The simple random forest classifier, with primarily default hyperparameters, performed very well with a OOB score of 0.960976 and a similar Kaggle score of 0.96342 (Account User ID 4810027)(https://www.kaggle.com/brianelinsky/). This model took 32 seconds to train, without parallelization. The PCA reduced the number of features to 332, and took 11 seconds to train. Feeding this into a random forest classifier produced an OOB score of 0.92252 and a Kaggle score of 0.93357. It took 73 seconds to train this second random forest classifier.

The issue with this experiment is the information leakage in the PCA. The PCA was fit with both training data and test data. Hence, the final model included some information from the test data. Next I replicated the experiment by fitting the PCA model with only training data, and using that model to transform the test data. Surprisingly this model performed no better than the model with information leakage. The OOB score was 0.92038, and the Kaggle score was 0.93114. This PCA took slightly less time to train, at 9 seconds. The random forest classifier was also slightly faster at 66 seconds.

MANAGEMENT RECOMMENDATIONS

I would not recommend spending more time applying dimensionality reduction models on the MNIST dataset. In some datasets, a significant proportion of the variance can be explained by the first 1-3 principal components. When that is the case, you can gain some understanding of the dataset by visualizing the first few components. However, I don't believe there is much business value in visualizing the first few components of a MNIST PCA. Additionally, I didn't get much benefit from using PCA as a tool to extract features.



```
>_ kaggle competitions submit -c digit-recognizer -f submission.csv -m "Message"
```

1 submissions for Brian Elinsky

Sort by Most recent

All Successful Selected

Submission and Description	Public Score	Use for Final Score
rf_clf_predictions.csv 9 hours ago by Brian Elinsky Fixed issue with last submission: This submission scales the test data before transforming it with the PCA	0.93114	☐
~~rf_clf_predictions.csv~~ ~~9 hours ago by Brian Elinsky~~ ~~PCA + RandomForestClassifier without the information leakage~~	0.44057	☐
rf_clf_predictions.csv 10 hours ago by Brian Elinsky THIS HAS A MISTAKE. I trained a PCA on the COMBINED test and training data. Then used the reduced features to train a RandomForestRegressor.	0.93357	☐
rf_clf_predictions.csv a day ago by Brian Elinsky Default hyperparameters for a RandomForestClassifier. No preprocessing.	0.96342	✓

No more submissions to show

runall.py

```python
# 2020-05-04 runall.py
from sklearn.ensemble import RandomForestClassifier
from lib_bre import *


# Load training data
data_file = get_dataset_file_path('2020-05-04', 'train.csv')
train = pd.read_csv(data_file)

# Remove label
X_train = train.drop(columns='label')
y_train = train['label'].copy()

rf_clf = RandomForestClassifier(random_state=98, oob_score=True)
rf_clf.fit(X_train, y_train)
score = rf_clf.oob_score_
print(rf_clf.get_params())
print(score)

# Evaluate final model on the test set
# Load test data
test_data_file = get_dataset_file_path('2020-05-04', 'test.csv')
X_test = pd.read_csv(test_data_file)
image_ids = pd.Series(data=range(1, 28001), name="ImageId")

# Make final predictions
rf_clf_predictions = rf_clf.predict(X_test)
rf_clf_predictions_series = pd.Series(rf_clf_predictions, name="Label")

# Output predictions
output = pd.concat([image_ids, rf_clf_predictions_series], axis=1)
output.to_csv("rf_clf_predictions.csv", index=False)
```

runall.py

```python
# 2020-05-05 runall.py
from sklearn.decomposition import PCA
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import StandardScaler
from lib_bre import *


# Load train data
train_data_file = get_dataset_file_path('2020-05-04', 'train.csv')
train = pd.read_csv(train_data_file)

# Remove label
X_train = train.drop(columns='label')
y_train = train['label'].copy()

# Load test data
test_data_file = get_dataset_file_path('2020-05-04', 'test.csv')
X_test = pd.read_csv(test_data_file)

# Combine test and train data (Don't copy this code)
combined_dataset = pd.concat([X_train, X_test], axis=0)
image_ids = pd.Series(data=range(1, 70001), name="ImageId")
combined_dataset.index = image_ids

# Create PCA model
X_combined_scaled = StandardScaler().fit_transform(combined_dataset)
pca = PCA(n_components=0.95, whiten=True, random_state=12)
X_combined_reduced = pca.fit_transform(X_combined_scaled)

print(pca.explained_variance_.shape)

X_train_reduced = X_combined_reduced[0:42000, :]

# Build Random Forest Classifier with reduced features
rf_clf = RandomForestClassifier(oob_score=True, random_state=12)
rf_clf.fit(X_train_reduced, y_train)
score = rf_clf.oob_score_
print(rf_clf.get_params())
print(score)

# Evaluate final model on the test set
# Load test data
X_test_reduced = X_combined_reduced[42000: , :]
image_ids = pd.Series(data=range(1, 28001), name="ImageId")

# Make final predictions
rf_clf_predictions = rf_clf.predict(X_test_reduced)
rf_clf_predictions_series = pd.Series(rf_clf_predictions, name="Label")

# Output predictions
output = pd.concat([image_ids, rf_clf_predictions_series], axis=1)
output.to_csv("rf_clf_predictions.csv", index=False)
```

runall.py

```python
# 2020-05-06 runall.py
from sklearn.decomposition import PCA
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import StandardScaler
from lib_bre import *


# Load train data
train_data_file = get_dataset_file_path('2020-05-04', 'train.csv')
train = pd.read_csv(train_data_file)

# Remove label
X_train = train.drop(columns='label')
y_train = train['label'].copy()

# Create PCA model
scaler = StandardScaler()
scaler.fit(X_train)
X_train_scaled = scaler.transform(X_train)
pca = PCA(n_components=0.95, whiten=True, random_state=12)
X_train_reduced = pca.fit_transform(X_train_scaled)

print(pca.explained_variance_.shape)

# Build Random Forest Classifier with reduced features
rf_clf = RandomForestClassifier(oob_score=True, random_state=12)
rf_clf.fit(X_train_reduced, y_train)
score = rf_clf.oob_score_
print(rf_clf.get_params())
print(score)

# Evaluate final model on the test set
# Load test data
test_data_file = get_dataset_file_path('2020-05-04', 'test.csv')
X_test = pd.read_csv(test_data_file)

# Scale the test data
X_test_scaled = scaler.transform(X_test)

# Reduce test data with PCA
X_test_reduced = pca.transform(X_test_scaled)

# Make final predictions
rf_clf_predictions = rf_clf.predict(X_test_reduced)
rf_clf_predictions_series = pd.Series(rf_clf_predictions, name="Label")

# Output predictions
image_ids = pd.Series(data=range(1, 28001), name="ImageId")
output = pd.concat([image_ids, rf_clf_predictions_series], axis=1)
output.to_csv("rf_clf_predictions.csv", index=False)
```

Digit Recognizer Lab Notebook

2020-05-04

- Define the objective in business terms.
    - The objective is to classify as many digits correctly as possible.
- How will your solution be used?
    - This is not stated in the problem.
- What are the current solutions/workarounds (if any)?
    - Presumably the current solution is a manual one, and we are looking to automate it.
- How should you frame this problem (supervised/unsupervised, online/offline, etc.)?
    - This is an offline supervised learning problem.
- How should performance be measured?
    - Performance will be measured by categorization accuracy.
- Is the performance measure aligned with the business objective?
    - Not sure, as the business objective is not stated.
- What would be the minimum performance needed to reach the business objective?
    - I think above 90% would be pretty good. And above 97% would be very good. http://yann.lecun.com/exdb/mnist/
- What are comparable problems? Can you reuse experience or tools?
    - I haven't solved any comparable problems so far.
- Is human expertise available?
    - Yes, I am an expert (we all are) in handwriting recognition.
- How would you solve the problem manually?
    - I would solve this visually, not looking at the matrix of data. This indicates to me that the current matrix may not be the best way to represent the data. I may need to transform it before I train a model.
- List the assumptions you (or others) have made so far.
    - The data is labeled correctly.
    - The numbers are relatively centered.
- Verify assumptions if possible.

Experiment Plan

This week I've been given explicit directions to do the following:

Begin by fitting a random forest classifier using the full set of 784 explanatory variables and the model training set (train.csv). Record the time it takes to fit the model and then evaluate the model on the test.csv data by submitting to Kaggle.com. Provide your Kaggle.com score and user ID.

Experiment Results

RandomForestClassifier parameters:

```
{'bootstrap': True, 'ccp_alpha': 0.0, 'class_weight': None, 'criterion': 'gini',
'max_depth': None, 'max_features': 'auto', 'max_leaf_nodes': None, 'max_samples':
None, 'min_impurity_decrease': 0.0, 'min_impurity_split': None,
'min_samples_leaf': 1, 'min_samples_split': 2, 'min_weight_fraction_leaf': 0.0,
'n_estimators': 100, 'n_jobs': None, 'oob_score': True, 'random_state': 98,
'verbose': 0, 'warm_start': False}
```

OOB Score: 0.9609761904761904

Kaggle Score: 0.96342 (Account User ID 4810027)(https://www.kaggle.com/brianelinsky/)

Time: 32.018 seconds (without parallelization)

2020-05-05

Experiment Plan

My instructions are as follows:

Execute principal components analysis (PCA) on the combined training and test set data together, generating principal components that represent 95 percent of the variability in the explanatory variables. The number of principal components in the solution should be substantially fewer than the 784 explanatory variables. Record the time it takes to identify the principal components.

Using the identified principal components, use the train.csv to build another random forest classifier. Record the time it takes to fit the model and to evaluate the model on the test.csv data by submitting to Kaggle.com. ***Provide your Kaggle.com score and user ID.***

Experiment Results

The PCA reduced the number of features to 332.

RandomForestClassifier parameters:

```
{'bootstrap': True, 'ccp_alpha': 0.0, 'class_weight': None, 'criterion': 'gini',
'max_depth': None, 'max_features': 'auto', 'max_leaf_nodes': None, 'max_samples':
None, 'min_impurity_decrease': 0.0, 'min_impurity_split': None,
'min_samples_leaf': 1, 'min_samples_split': 2, 'min_weight_fraction_leaf': 0.0,
'n_estimators': 100, 'n_jobs': None, 'oob_score': True, 'random_state': 12,
'verbose': 0, 'warm_start': False}
0.9225238095238095
```

OOB Score: 0.9225238095238095

Kaggle Score: 0.93357 (Account User ID 4810027)(https://www.kaggle.com/brianelinsky/)

Time:

- 11.166 seconds to train the PCA
- 73.140 seconds (without parallelization) to train the Random Forest Classifier

The issue with this experiment is the information leakage in the PCA. The PCA was fit with both training data and test data. Hence, the final model included some information from the test data. This experiment should be replicated by fitting the PCA model with only training data, and using that model to transform the test data. There was also information leakage in the scaler.

I was surprised that the OOB score and the Kaggle score were so close. I expected the Kaggle score to be lower due to the information leakage.

2020-05-06

Experiment Plan

Repeat the same experiment but without the information leakage.

The experiment we have proposed has a MAJOR design flaw. Identify the flaw. Fix it. Rerun the experiment in a way that is consistent with a training-and-test regimen, and submit this to Kaggle.com. Provide your Kaggle.com score and user ID

Experiment Results

The PCA reduced the number of features to 320.

RandomForestClassifier parameters:

```
{'bootstrap': True, 'ccp_alpha': 0.0, 'class_weight': None, 'criterion': 'gini',
'max_depth': None, 'max_features': 'auto', 'max_leaf_nodes': None, 'max_samples':
None, 'min_impurity_decrease': 0.0, 'min_impurity_split': None,
'min_samples_leaf': 1, 'min_samples_split': 2, 'min_weight_fraction_leaf': 0.0,
'n_estimators': 100, 'n_jobs': None, 'oob_score': True, 'random_state': 12,
'verbose': 0, 'warm_start': False}
```

OOB Score: 0.9203809523809524

Kaggle Score: 0.93114 (Account User ID 4810027)(https://www.kaggle.com/brianelinsky/)

Time:

- 8.834 seconds to train the PCA
- 66.2 seconds (without parallelization) to train the Random Forest Classifier