practical_ml

August 10, 2019

0.1 Practical Machine Learning Assignment

```
In [1]: from sklearn.multiclass import OneVsOneClassifier, OneVsRestClassifier
        from sklearn.metrics import classification_report
        from sklearn.tree import DecisionTreeClassifier
        from sklearn.model_selection import cross_val_score
        from sklearn.preprocessing import LabelEncoder
        from collections import defaultdict
        import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
```

0.2 Summary of Prediction

Note: I used Python, not R

In this project, I used a Decision Tree classifier into a one-vs-one multiclass classifier. The reason I choosed a Tree-based model is because of the large number of feature in the data. Using a linear model sush as Logistic Regression or Support Vector Machine would have required one-hot encoing which would have increased the dimension even further. We do not have that problem with Tree-Based models, as label encoding works just fine.

I started simple with a Decision Tree and it turned out to be the best model. I found that random forest overfitted the train set even with a small number of Trees in the forest. Also, I had to drop many features which had more than 50% missing values, which reduced the dimention from 160 to 60.

I used 10-fold cross validation to estimate the out-of-sample accuracy on the test set.

0.2.1 Data Import and Exploration

```
In [2]: df = pd.read_csv('pml-training.csv', low_memory=False)
```

```
In [3]: print(df.shape)
        df[df.columns[0:10]].head()
```

```
(19622, 160)
```

```
Out[3]:    Unnamed: 0 user_name  raw_timestamp_part_1  raw_timestamp_part_2  \
        0           1  carlitos            1323084231                788290
```

```
1           2  carlitos           1323084231                808298
2           3  carlitos           1323084231                820366
3           4  carlitos           1323084232                120339
4           5  carlitos           1323084232                196328

     cvtd_timestamp new_window  num_window  roll_belt  pitch_belt  yaw_belt
0  05/12/2011 11:23         no          11       1.41        8.07     -94.4
1  05/12/2011 11:23         no          11       1.41        8.07     -94.4
2  05/12/2011 11:23         no          11       1.42        8.07     -94.4
3  05/12/2011 11:23         no          12       1.48        8.05     -94.4
4  05/12/2011 11:23         no          12       1.48        8.07     -94.4
```

In [4]:
```
y = df['classe']
df_clean = df.drop(['Unnamed: 0', 'classe'], axis=1)
```

In [5]:
```
#find columns with large number of missing values
missing = []
for column in df_clean.columns:
    if df_clean[column].isna().sum()/df_clean.shape[0] > 0.5:
        missing.append(column)
df_clean = df_clean.drop(missing, axis=1)

objects = df_clean.select_dtypes(include=['object'])
numerics = df_clean.select_dtypes(include=['int', 'float64', 'int64'])
print(len(objects.columns) + len(numerics.columns) == len(df_clean.columns))
print(objects.shape, numerics.shape)
```

```
True
(19622, 3) (19622, 55)
```

In [6]:
```
d = defaultdict(LabelEncoder)
categoric = objects.apply(lambda x: d[x.name].fit_transform(x))
```

In [7]:
```
X_train = pd.concat([categoric, numerics], axis=1, sort=False)
y_train = LabelEncoder().fit_transform(y)
```

In [8]:
```
predictor = OneVsOneClassifier(DecisionTreeClassifier(random_state = 1, max_depth = 5))
y_pred = predictor.fit(X_train, y_train).predict(X_train)
target_names = ['class 0', 'class 1', 'class 2', 'class 3', 'class 4']
print(classification_report(y_train, y_pred, target_names=target_names))
```

```
              precision    recall  f1-score   support

     class 0       0.93      0.89      0.91      5580
     class 1       0.90      0.78      0.83      3797
     class 2       0.86      0.92      0.89      3422
     class 3       0.78      0.90      0.83      3216
     class 4       0.93      0.94      0.93      3607
```

```
   micro avg       0.88      0.88      0.88     19622
   macro avg       0.88      0.88      0.88     19622
weighted avg       0.89      0.88      0.88     19622
```

0.2.2 Esimating Out-of-Sample Error with Cross-Validation Score

Given the above results, I expect out-of-sample error to be significant. The above model will probably not generalize well to new data because the variance appears to be high. Let's use cross-validation to estimate out of sample error.

```
In [9]: scores = cross_val_score(predictor, X_train, y_train, cv = 10)
        print("Mean Cross-Validation Score: ", np.mean(scores))
```

```
Mean Cross-Validation Score:  0.6595557261538972
```

0.2.3 Test Set Performance

```
In [10]: df_test = pd.read_csv('pml-testing.csv', low_memory=False).drop(['Unnamed: 0'], axis=
         print(df_test.shape)
         df_test = df_test.drop(missing, axis=1)
         print(df_test.shape)
         objects_df = df_test.select_dtypes(include=['object'])
         numerics_df = df_test.select_dtypes(include=['int', 'float64', 'int64'])
         print(objects_df.shape, numerics_df.shape)
         print(len(objects_df.columns) + len(numerics_df.columns) == len(df_test.columns))

         categoric_df = objects_df.apply(lambda x: d[x.name].transform(x))
         X_test = pd.concat([categoric_df, numerics_df], axis=1, sort=False)[X_train.columns]

         y_test = predictor.predict(X_test)
```

```
(20, 159)
(20, 59)
(20, 3) (20, 56)
True
```

```
In [11]: y_test
```

```
Out[11]: array([1, 0, 2, 0, 0, 4, 3, 3, 0, 0, 1, 2, 1, 0, 4, 4, 0, 3, 1, 1])
```

```
In [13]: y.unique()
```

```
Out[13]: array(['A', 'B', 'C', 'D', 'E'], dtype=object)
```

In [15]:
```
dic = {0:'A', 1:'B', 2:'C', 3:'D', 4:'E'}
results = pd.DataFrame({'prediction': y_test, 'classe': [dic[i] for i in y_test]})
results
```

Out[15]:

	prediction	classe
0	1	B
1	0	A
2	2	C
3	0	A
4	0	A
5	4	E
6	3	D
7	3	D
8	0	A
9	0	A
10	1	B
11	2	C
12	1	B
13	0	A
14	4	E
15	4	E
16	0	A
17	3	D
18	1	B
19	1	B