Michael Alvarez
Madelon Dataset

Task

Our task was to develop a series of notebooks and models for the purpose of identifying relevant features with different methods and generating predictions through multiple different models

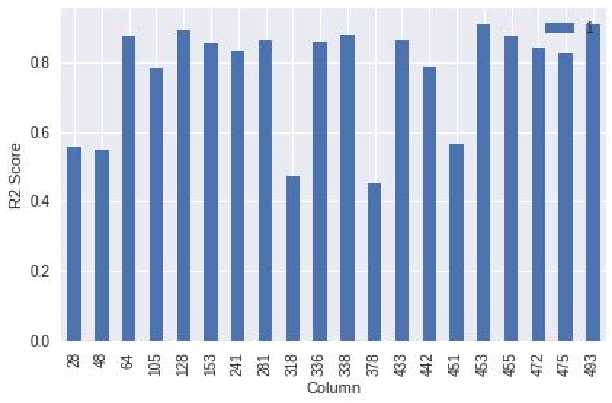
We begin with the UCI dataset:

“MADELON is an artificial dataset containing data points grouped in 32 clusters placed on the vertices of a five dimensional hypercube and randomly labeled +1 or -1. The five dimensions constitute 5 informative features. 15 linear combinations of those features were added to form a set of 20 (redundant) informative features.” Based on those 20 features one must separate the examples into the 2 classes (corresponding to the +-1 labels). We added a number of distractor feature called 'probes' having no predictive power. The order of the features and patterns were randomized. “ - UCI site

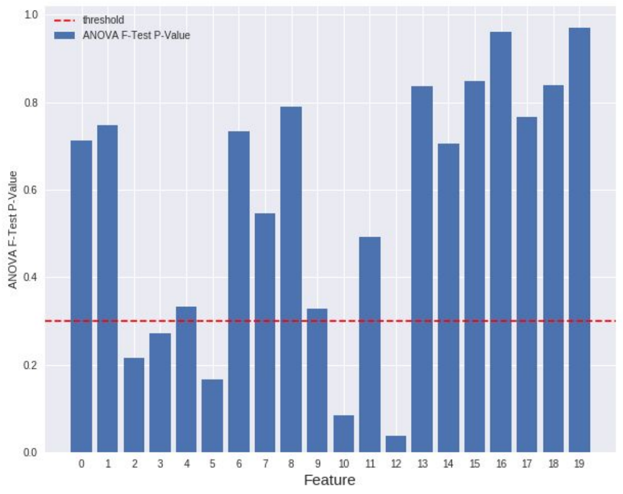
My first step after some really light EDA was to eliminate the noise in the data. This was by means of calculating the R2 score. The purpose of which is finding which features have high correlation. The reason this give us our salient features is because we know there are 20 features and that for the most part are linear combinations of each other

This gave us:

.



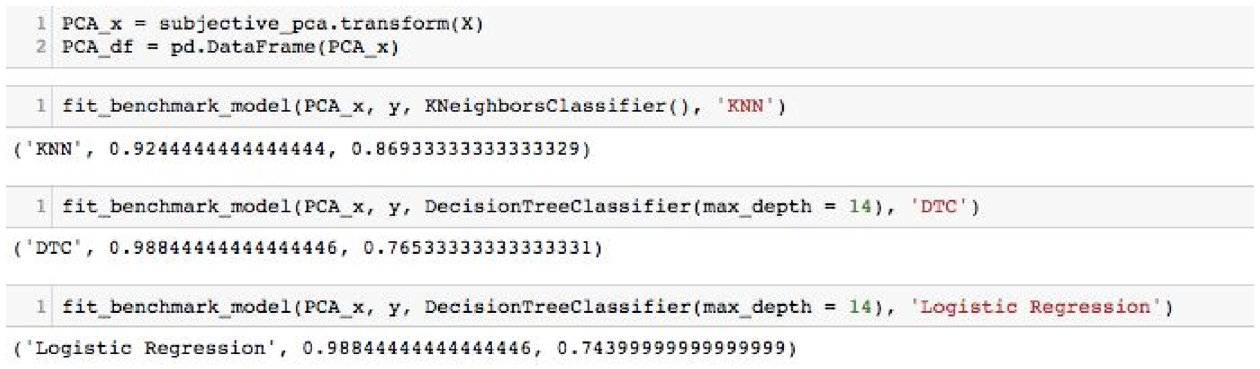
We can also use the Anova F-test: this gave us pretty much the same result as the R2 method.



BenchMarking:

We fit 3 different benchmarks, A decision tree, a logistic regression, and K-nearest neighbors. K-nearest neighbors performed the best, classifying the data correctly 85% of the time.

PCA



```
PCA_x = subjective_pca.transform(X)
PCA_df = pd.DataFrame(PCA_x)
```

```
fit_benchmark_model(PCA_x, y, KNeighborsClassifier(), 'KNN')
```

```
('KNN', 0.9244444444444444, 0.86933333333333329)
```

```
fit_benchmark_model(PCA_x, y, DecisionTreeClassifier(max_depth = 14), 'DTC')
```

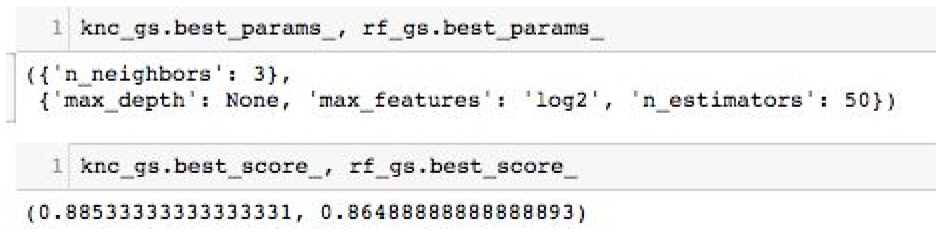
```
('DTC', 0.98844444444444446, 0.76533333333333331)
```

```
fit_benchmark_model(PCA_x, y, DecisionTreeClassifier(max_depth = 14), 'Logistic Regression')
```

```
('Logistic Regression', 0.98844444444444446, 0.74399999999999999)
```

Once again KNN proves to be the best with 86% of the data successfully classified correctly

Next we run a gridsearch for each model.



```
knc_gs.best_params_, rf_gs.best_params_
```

```
({'n_neighbors': 3},
 {'max_depth': None, 'max_features': 'log2', 'n_estimators': 50})
```

```
knc_gs.best_score_, rf_gs.best_score_
```

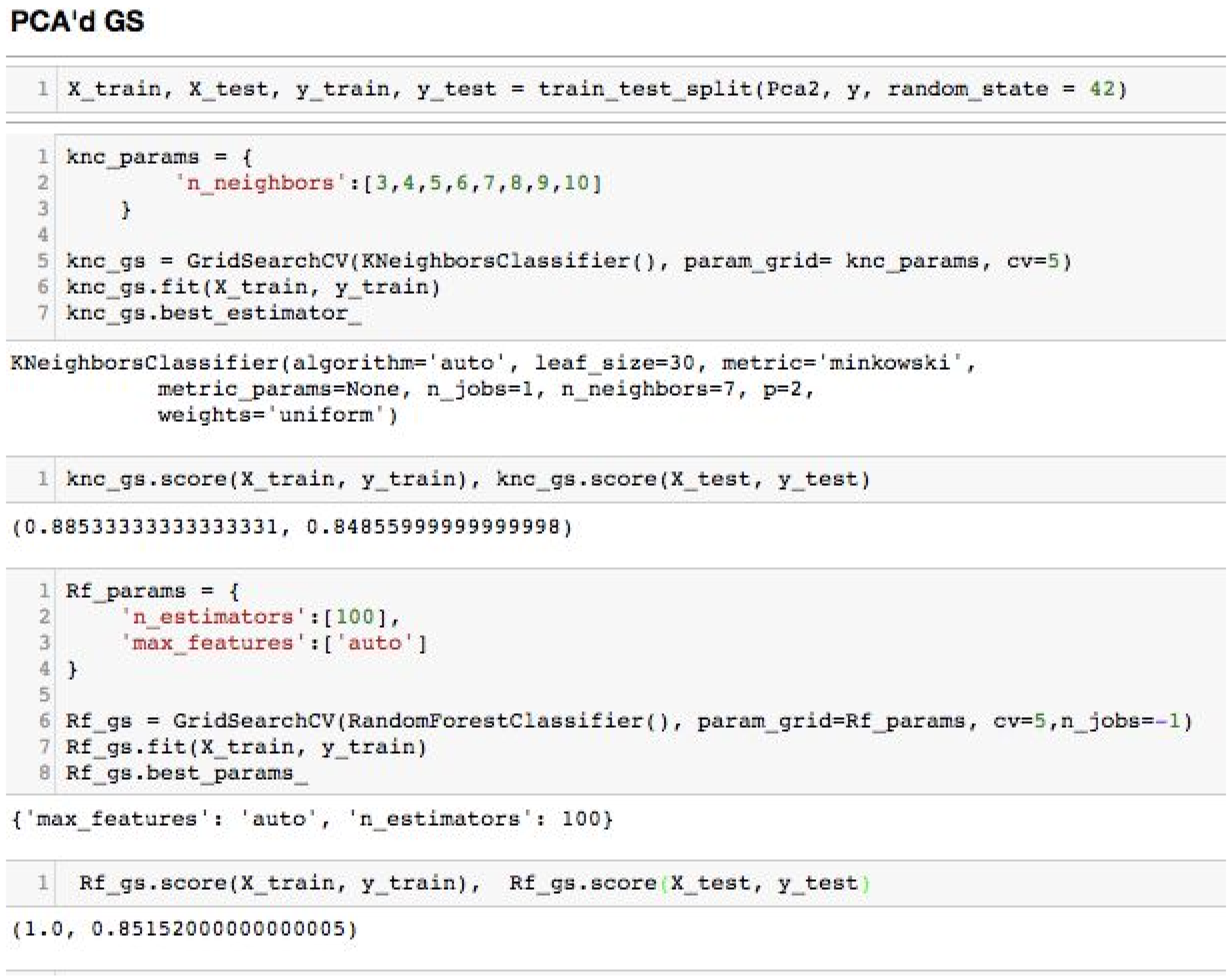
```
(0.88533333333333331, 0.86488888888888893)
```

After all is said a running these KNN is still the performs best with 88%.

Instructor Data

This essentially follows the same work flow.

After connecting to the database and load the data, we run the same models. Below are the final results:



PCA'd GS

```
X_train, X_test, y_train, y_test = train_test_split(Pca2, y, random_state = 42)
```

```
knc_params = {
        'n_neighbors':[3,4,5,6,7,8,9,10]
    }

knc_gs = GridSearchCV(KNeighborsClassifier(), param_grid= knc_params, cv=5)
knc_gs.fit(X_train, y_train)
knc_gs.best_estimator_
```

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
           metric_params=None, n_jobs=1, n_neighbors=7, p=2,
           weights='uniform')
```

```
knc_gs.score(X_train, y_train), knc_gs.score(X_test, y_test)
```

```
(0.88533333333333331, 0.84855999999999998)
```

```
Rf_params = {
    'n_estimators':[100],
    'max_features':['auto']
}

Rf_gs = GridSearchCV(RandomForestClassifier(), param_grid=Rf_params, cv=5,n_jobs=-1)
Rf_gs.fit(X_train, y_train)
Rf_gs.best_params_
```

```
{'max_features': 'auto', 'n_estimators': 100}
```

```
 Rf_gs.score(X_train, y_train),  Rf_gs.score(X_test, y_test)
```

```
(1.0, 0.85152000000000005)
```

From the instructor dataset the Random FOrest Classifier performed best with 85%.