Project 3: Madelon Feature Selection and Classification

Author: Uyen Truong

Project Description:

This project, I will be working on the Madelon dataset. One set of data is downloaded from the UCI website and the other artificial dataset was queried from a Database. The dataset from UCI has 500 features and of these 500 features, only 20 are real, where 5 are informative, the other 15 are linear combinations of the 5 informative features. The dataset from the Database has 1000 features and 220,000 rows of data, no other information was given about this dataset in regards to the number of informative features.

The goal of the project is to use statistical models for feature slection and classification in order to best predict the examples into 2 classes corresponding to the target values (-1 or 1 for UCI dataset and 0 or 1 for Database dataset).

Initial Step: Download the data and perform EDA

Although the instructions indicated that we needed to work on three 10% sets of the database dataset, computing such large datasets of about 22,000 rows and 1000 columns is computational intensive and constantly crashed AWS T-2Micro machine. Therefore, I used Survey Monkey Sample Size calculator to determine a sample size that would be suitable for analysis. I determined that a sample size that gives me 95% confidence interval and 1% margin error would be sufficient for analysis. According to Survey Monkey Calculator, this calls for about 5% of the dataset which is equivalent to about 9200-9500 rows of the Database dataset. All of the calculations for the database dataset were run on three sets of 5% of the data.

Initial EDA of the datasets indicated that there are no null values in the data except for one column in the UCI dataset where all the values were NaN, so I dropped that column. A heatmap of the entire UCI dataset and a sample of the Database dataset show that there are significant noise in both sets and it is difficult to determine any correlation from looking at the map.

A heatmap of the entire UCI dataset and a sample of the Database dataset show that there are significant noise in both sets and it is difficult to determine any correlation from looking at the map.

I alsocreated scatter plots of some of the features to see if there are any correlations but it was difficult to see if relationship between the features.

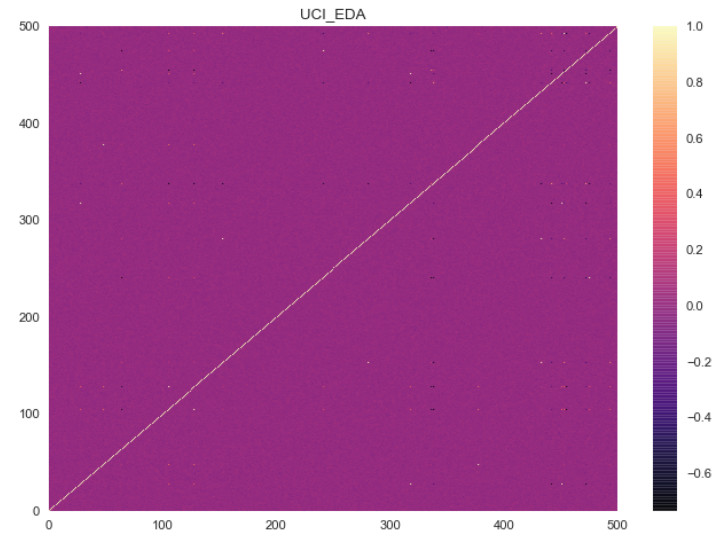


Figure 1:

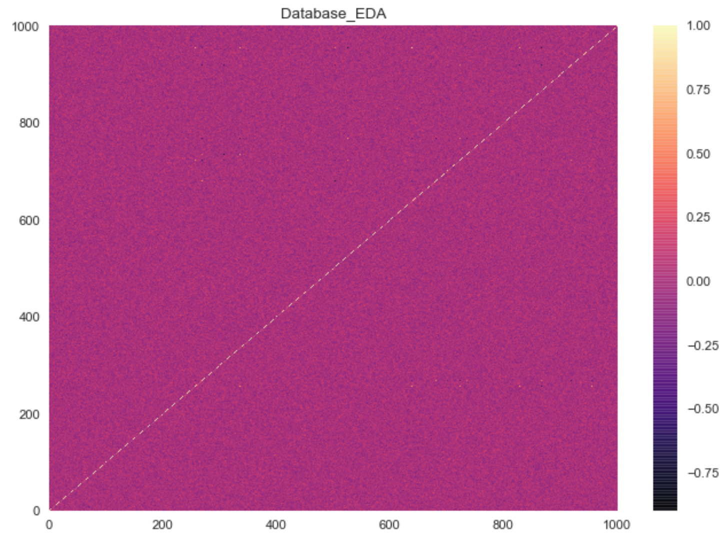
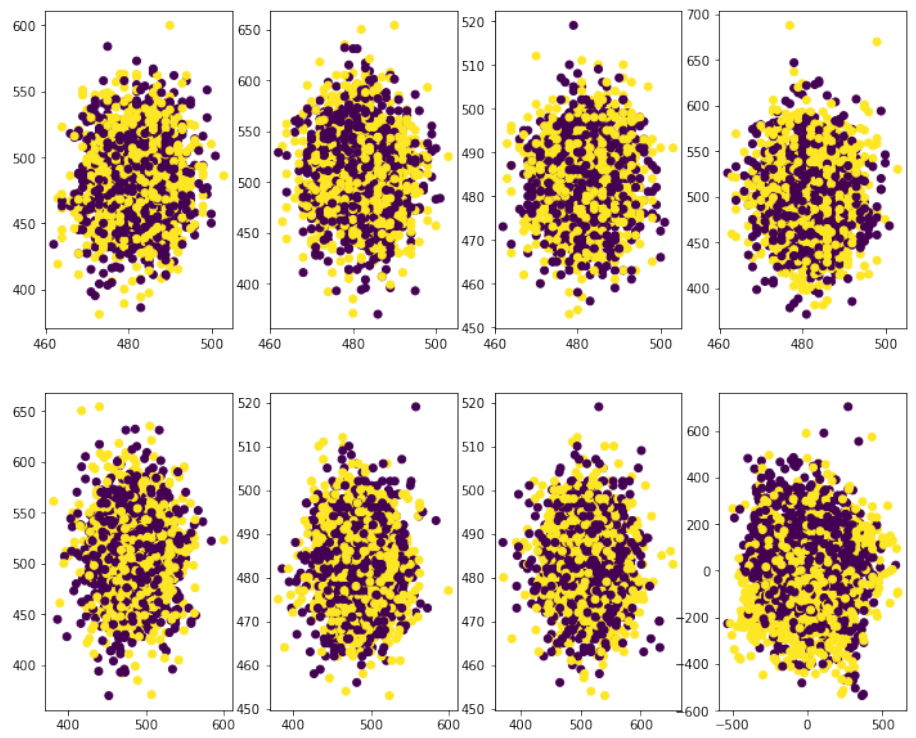


Figure 2:



Step 1: Benchmarking

I built pipelines to perform naive fits to set the benchmark scores with the entire UCI train dataset and 5% of Database dataset with the following models:

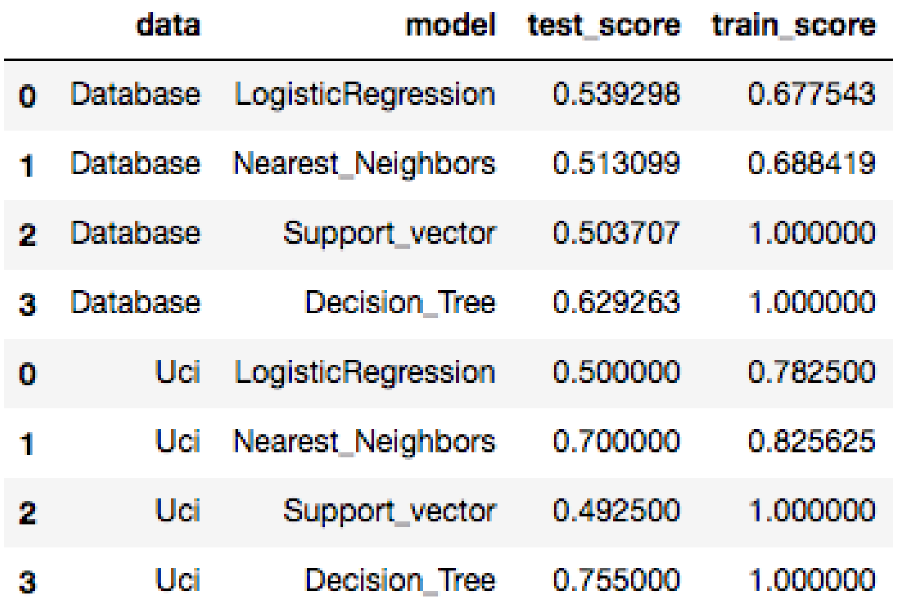
- Logistic Regression(LR)
- Decision Tree(Tree)
- K Nearest Neighbors(KNN)
- Support Vector Classier(SVC)

According to the project instructions, a large C must be set for Logistic Regression and Support Vector Classier to minimize regulization. Therefore, I set the C value equal to 1e10 for both LR and SVC. Below are the results of the benchmark scores.

Note: the scores present throughout this notebook are Accuracy scores.

Step 2: Finding Salient Features

Given the information about the UCI Madelon dataset where there are 480 'noisy' features out of the 500, I used the feature selection methods described



	data	model	test_score	train_score
0	Database	LogisticRegression	0.539298	0.677543
1	Database	Nearest_Neighbors	0.513099	0.688419
2	Database	Support_vector	0.503707	1.000000
3	Database	Decision_Tree	0.629263	1.000000
0	Uci	LogisticRegression	0.500000	0.782500
1	Uci	Nearest_Neighbors	0.700000	0.825625
2	Uci	Support_vector	0.492500	1.000000
3	Uci	Decision_Tree	0.755000	1.000000

Figure 3:

below to identify the 20 relevant features. While no information was given for the Madelon Database dataset, I assumed that it would have similar structure as the UCI dataset where there are many 'noisy' features, some redundant linear combinations and some informative features. Therefore, I applied the same methods on the Database dataset.

Method 1: Calculate R^2 scores

In this method, the goal is to remove the actual target column and calculate the Rˆ2 scores of each feature taking turn as the target. This technique can tell us if there's a linear relationship between the features. If the Rˆ2 is high, that means there is a linear relationship with the other features; therefore, it could be one of the informative or redudant features. If the R^2 is low, that means it is independent of the all other features and would most likely not be a redundant or informative feature.

In this method, KneighborRegressor and DecisionTreeRegressor were used to calculate the R^2 scores. **Important note**: each time the R^2 was calculated for each feature, it yielded similar but different results; therefore, the R^2 was calculated multiple times and the mean R^2 was used to determine which features could be one of the salient features.

Method 1: Results

For the UCI dataset, the results of the top R^2 of both KneighborsRegressor and DecisionTree were identical. This method identified 20 relevant features: 28, 48, 64, 105, 128, 153, 241, 281, 318, 336, 338, 378, 433, 442, 451, 453, 455, 472, 475, and 493.

For the Database dataset, I only ran the code for DecisionTree since this dataset is much larger and the computational time would be too intensive. It can be inferred from the UCI dataset that the results for DecisionTree and KneighborRegressor would be the same. This method also identified 20 relevant features for the Database dataset:'feat_257', 'feat_269', 'feat_308', 'feat_315', 'feat_336', 'feat_341','feat_395', 'feat_504', 'feat_526', 'feat_639', 'feat_681', 'feat_701','feat_724', 'feat_736', 'feat_769', 'feat_808', 'feat_829', 'feat_867','feat_920', 'feat_956'.

Method 2: Find top 20 features from SelectKBest

In this method, the goal is to find the top 20 features that would predict the target using SelectKbest pipeline with the score_function = f_regression.

Method 2 Results:

Unfortunately, the top 20 features identified by SelectKBest did not return the same 20 features as the R^2 method; however, it did generate some overlaps.

For the Uci sample dataset, the top 20 features identified were: 48, 64, 105, 128, 137, 149, 199, 204, 241, 282, 329, 336, 338, 378, 424, 442, 453, 472, 475, 493. Although not all 20 match with the R^2 method, 13 out of 20 features from SelectKBest were the same as the ones from R^2.

For the Database sample dataset, the top 20 features identified were: 'feat_003', 'feat_086', 'feat_257', 'feat_269', 'feat_308', 'feat_315','feat_336', 'feat_341', 'feat_395', 'feat_504', 'feat_526', 'feat_557','feat_597', 'feat_673', 'feat_681', 'feat_691', 'feat_701', 'feat_724', 'feat_736', 'feat_769', 'feat_783', 'feat_808', 'feat_829', 'feat_867','feat_920'. Although not all 20 match with the R^2 method, 17 out of 20 features from SelectKBest were the same as the ones from R^2.

Method 3: Generate correlation matrix

In this method, the goal is to create a correlation matrix of all the features and find the ones with strong correlations to other features. This method is similar to the R^2 method where the Informative and Redundant features will have stronger correlations whereas the noisy features will have weak correlations.

Method 3: Results

For the UCI dataset, this method identified the same 20 features with high correlationthat were identified by R^2 method.

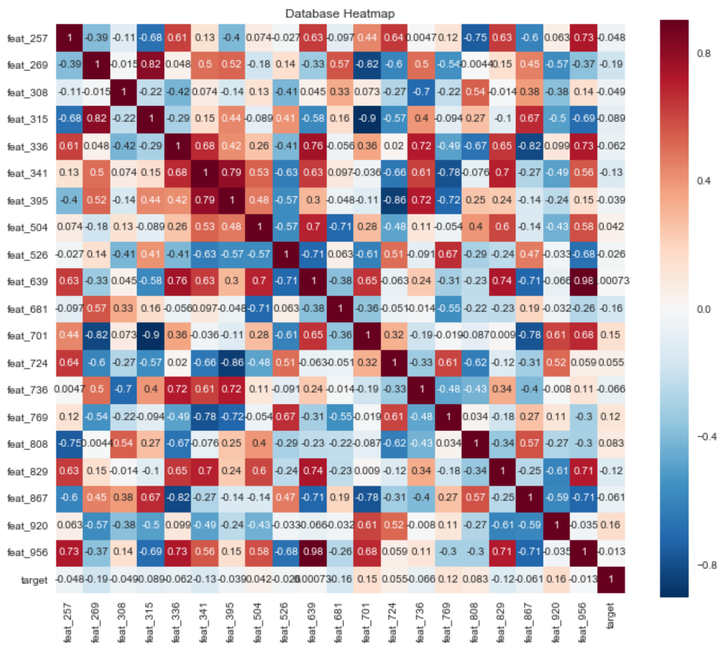
For the Database dataset, this method also identified the same 20 features with high correlation that were identified by the R^2 method.

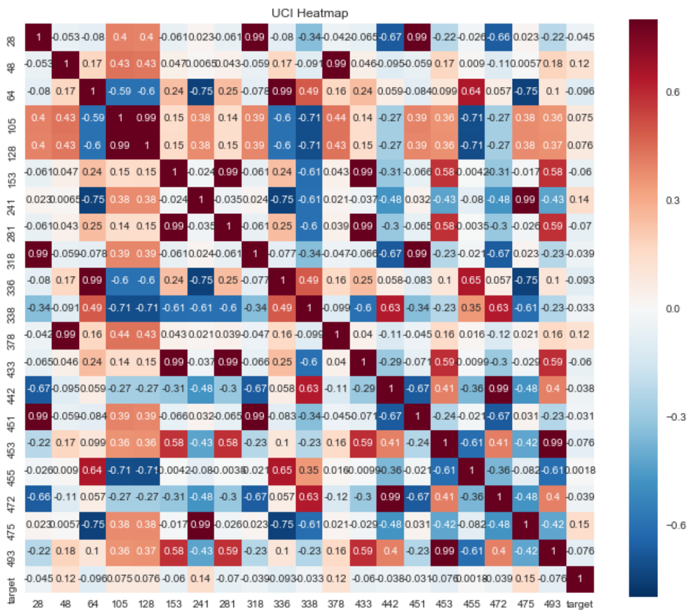
Feature Selection Conclusions:

Even though I did not get consistent results across all three methods, R^2 and Correlation Matrix identified the same top 20 features for the UCI and for Database datasets. Therefore, these will be the 20 features that I will use to build models for classifications.

Step 3: Feature Importance

The goal of this step is to identify the 5 important features that can be used to classify the target. I used three different methods on three sample UCI datasets and three 5% samples of the Database dataset. Since the top 20 salient features were identified in the previous step, I created new dataframes of the UCI and Database datasets to only include those 20 features in order to reduce the size of the datasets which will make computing time faster. I initially looked at the heatmap of the top 20 features to see their correlations to the target. As you can see in the figures below, there is no one feature that is strongly correlated to the target column. This is because some features are highly correlated with each other and no one feature is strongly correlated to the target.


Database Heatmap
feat_257
feat_269
feat_308
feat_315
feat_336
feat_341
feat_395
feat_504
feat_526
feat_639
feat_681
feat_701
feat_724
feat_736
feat_769
feat_808
feat_829
feat_867
feat_920
feat_956
target
0.8
0.4
0.0
−0.4
−0.8


UCI Heatmap
28
48
64
105
128
153
241
281
318
336
338
378
433
442
451
453
455
472
475
493
target
0.6
0.3
0.0
−0.3
−0.6

I used the method below in order to eliminate redundant features.

Method 1: SelectKbest to identify top 5 features

In this method, the goal is to find the top 5 features that would predict the target using SelectKbest pipeline with the score_function = f_regression.

Method 1: Results

As you can see in the results below, the top 5 features found in each of the 3 UCI sample datasets were inconsistent. The results from the database looked more promising! All 3 sample sets identified the same top 5 features.

- **UCI SKB results:**
  - UCI skb sample 1 result: 48, 128, 241, 378, 475
  - UCI skb sample 2 result: 64, 105, 241, 336, 475
  - UCI skb sample 3 result: 241, 336, 338, 472, 475
- **Database SKB results:**
  - Database skb sample 1 result:'feat_269','feat_341','feat_681','feat_701','feat_920'
  - Databaes skb sample 2 result: 'feat_269','feat_341','feat_681','feat_701','feat_920'
  - Database skb sample 3 result: 'feat_269','feat_341','feat_681','feat_701','feat_920'

Method 2: Recursive Feature Elimination to identify top 5 features**
In this method, the goal is to find the top 5 features that would predict the target using RFE with DecisionTree as the estimator.

Method 2: Results

As you can see in the results below, the top 5 features found in each of the 3 UCI sample datasets were inconsistent and the top 5 features found in each of the 3 Database datasets were also inconsistent. These results also varied from the results found in Method 1.

- **UCI RFE results:**
  - UCI RFE sample 1 result: 105, 318, 336, 378, 455
  - UCI RFE sample 2 result: 48, 105, 153, 451, 475
  - UCI RFE sample 3 result: 64, 105, 318, 338, 475
- **Database RFE results:**
  - Database RFE sample 1 result:'feat_269','feat_639','feat_769','feat_808','feat_920'
  - Databaes RFE sample 2 result: 'feat_269','feat_724','feat_736','feat_769','feat_829'
  - Database RFE sample 3 result: 'feat_269','feat_736','feat_769','feat_808','feat_829'

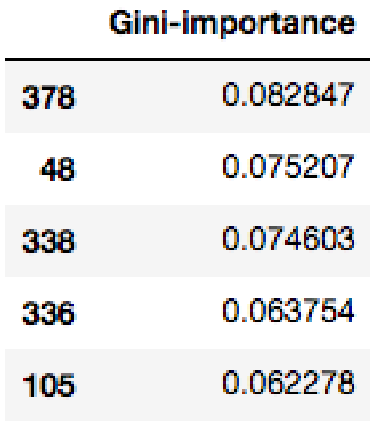
Method 3: Feature_importance with RandomForest

In this method, the goal is to find the top 5 features which have the highest Gini-Importance.

Method 3: Results

As you can see in the results below, the top 5 features found in each of the 3 UCI sample datasets were inconsistent from each other other. The top 5 features found in each of the Database sample datasets were also inconsistent. These results also varied from the results found in methods 1 and 2.

UCI Gini-Importance results:



	Gini-importance
378	0.082847
48	0.075207
338	0.074603
336	0.063754
105	0.062278

Figure 4:

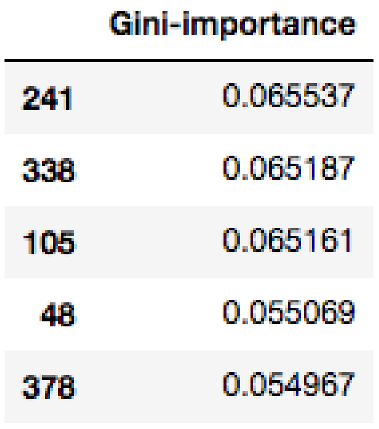
Database Feature Importance results:

Feature Importance Conclusions:

While there are some overlaps of features between the 3 methods, the results were inconclusive of which 5 features are the most important; therefore, I decided to keep all 20 salient features in model creation and to implement the feature reduction methods within the model pipelines.

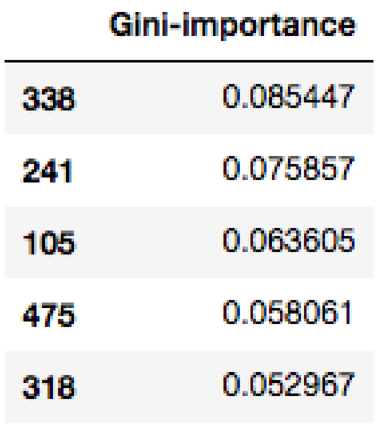
Step 4: Pipelines/GS to Build and Test Models

In this step, I created functions to run multiple types of pipelines and Gridsearch with different feature selection methods within the pipelines. I ran the Gridsearch



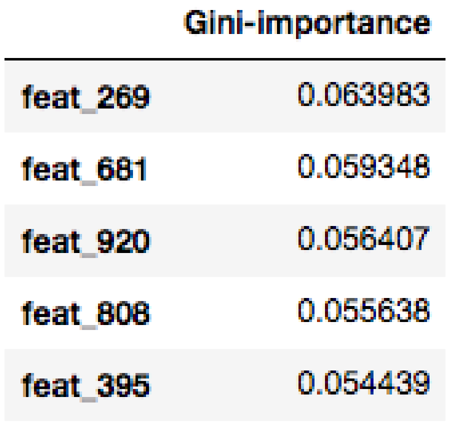
	Gini-importance
241	0.065537
338	0.065187
105	0.065161
48	0.055069
378	0.054967

Figure 5:



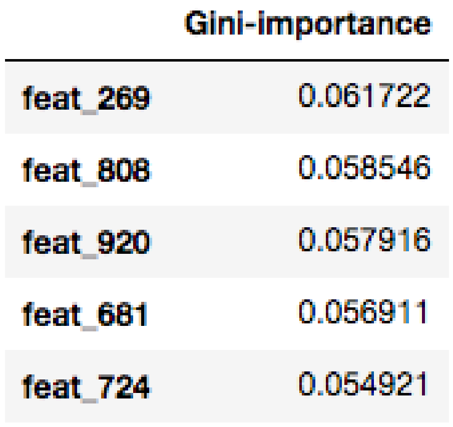
	Gini-importance
338	0.085447
241	0.075857
105	0.063605
475	0.058061
318	0.052967

Figure 6:



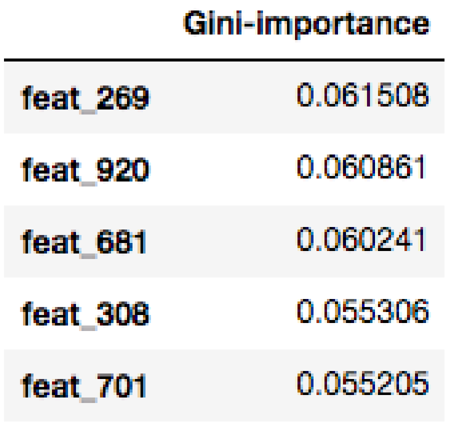
	Gini-importance
feat_269	0.063983
feat_681	0.059348
feat_920	0.056407
feat_808	0.055638
feat_395	0.054439

Figure 7:



	Gini-importance
feat_269	0.061722
feat_808	0.058546
feat_920	0.057916
feat_681	0.056911
feat_724	0.054921

Figure 8:



	Gini-importance
feat_269	0.061508
feat_920	0.060861
feat_681	0.060241
feat_308	0.055306
feat_701	0.055205

Figure 9:

for 6 sample datasets, 3 samples from UCI and 3 samples from the Database. The following Pipelines were created: - SelectKBest as Feature Reduction with the following classifier: - KNeighborsClassifier - LogisticRegression - DecisionTreeClassifier - RandomForestClassifier - SVC - SelectfromModel with Ridgeclassifier as an estimator ran in with the following classifier: - KNeighborsClassifier - LogisticRegression - DecisionTreeClassifier - RandomForestClassifier - SVC - PCA as an estimator ran in with the following classifier: - KNeighborsClassifier - LogisticRegression - DecisionTreeClassifier - RandomForestClassifier - SVC Each pipeline included specific parameters related to each steps. The codes below illustrate the sample pipelines and parameters that were used to create the pipelines (details for each Feature Reduction function and pipelines can be found in notebooks 04).

Please note: for the PCA method, data was only scaled and not deskewed. The sample graph below shows that the data was equally distributed within each of the 20 salient features (similar graphs for all 6 sample sets to can be found in notebooks 4).

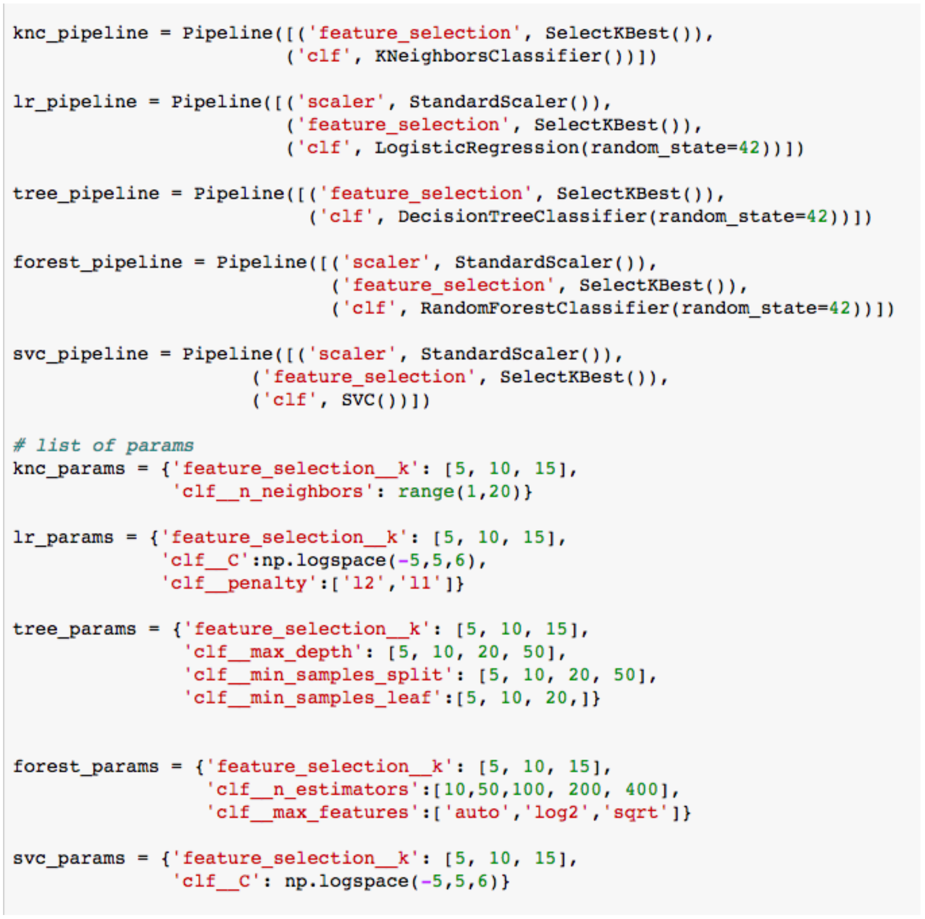
Gridsearch Results

For each sample dataset, there were 15 results from the pipeline/gridsearch. For the sake of space, only the top 5 scores of all results are shown below:

Database top 5 results:

UCI top 5 results:

As you can see, the best scores from both the Database and UCI were there following: - PCA method feature reduction method - SVC estimator - C = 10.0



```
knc_pipeline = Pipeline([('feature_selection', SelectKBest()),
                        ('clf', KNeighborsClassifier())])

lr_pipeline = Pipeline([('scaler', StandardScaler()),
                       ('feature_selection', SelectKBest()),
                       ('clf', LogisticRegression(random_state=42))])

tree_pipeline = Pipeline([('feature_selection', SelectKBest()),
                         ('clf', DecisionTreeClassifier(random_state=42))])

forest_pipeline = Pipeline([('scaler', StandardScaler()),
                           ('feature_selection', SelectKBest()),
                           ('clf', RandomForestClassifier(random_state=42))])

svc_pipeline = Pipeline([('scaler', StandardScaler()),
                     ('feature_selection', SelectKBest()),
                     ('clf', SVC())])

# list of params
knc_params = {'feature_selection__k': [5, 10, 15],
             'clf__n_neighbors': range(1,20)}

lr_params = {'feature_selection__k': [5, 10, 15],
            'clf__C':np.logspace(-5,5,6),
            'clf__penalty':['l2','l1']}

tree_params = {'feature_selection__k': [5, 10, 15],
              'clf__max_depth': [5, 10, 20, 50],
              'clf__min_samples_split': [5, 10, 20, 50],
              'clf__min_samples_leaf':[5, 10, 20,]}


forest_params = {'feature_selection__k': [5, 10, 15],
                'clf__n_estimators':[10,50,100, 200, 400],
                'clf__max_features':['auto','log2','sqrt']}

svc_params = {'feature_selection__k': [5, 10, 15],
             'clf__C': np.logspace(-5,5,6)}
```

Figure 10:

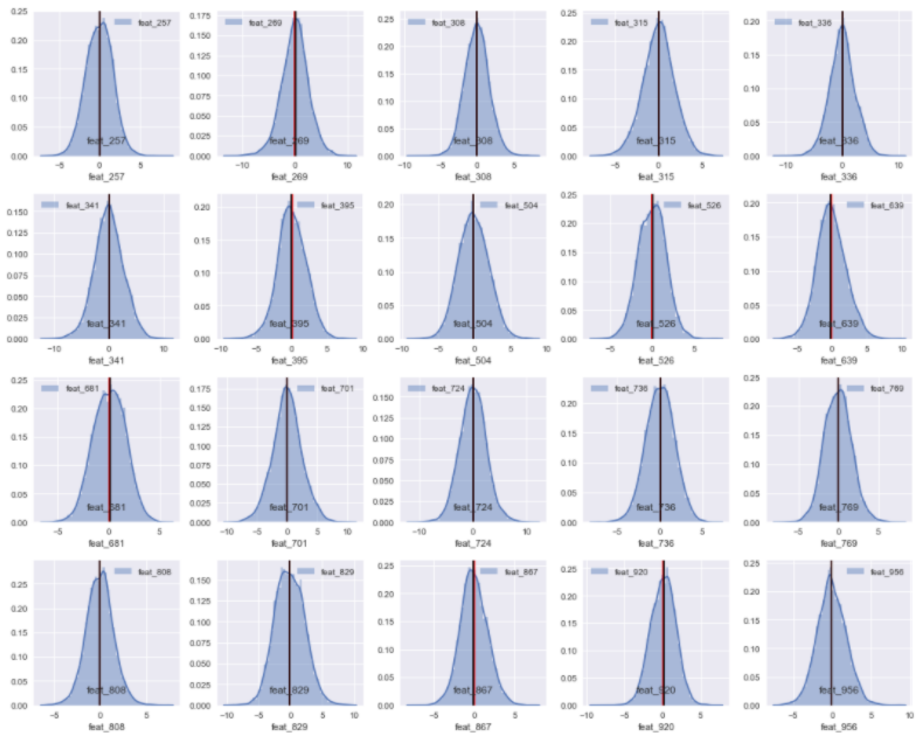
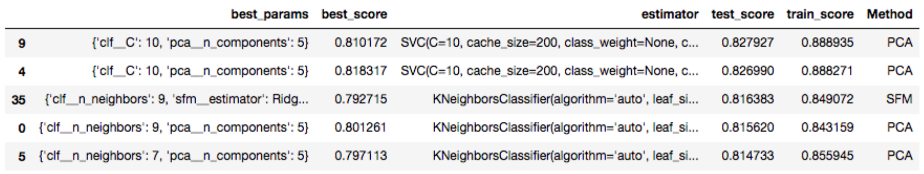
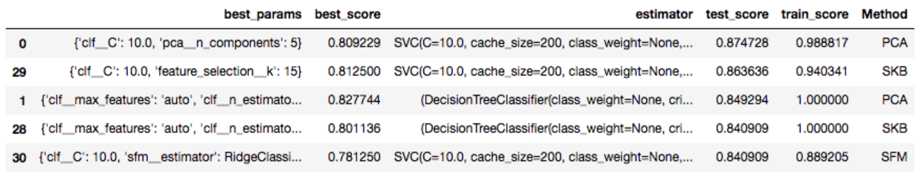


Figure 11:



	best_params	best_score	estimator	test_score	train_score	Method
9	{'clf__C': 10, 'pca__n_components': 5}	0.810172	SVC(C=10, cache_size=200, class_weight=None, c...	0.827927	0.888935	PCA
4	{'clf__C': 10, 'pca__n_components': 5}	0.818317	SVC(C=10, cache_size=200, class_weight=None, c...	0.826990	0.888271	PCA
35	{'clf__n_neighbors': 9, 'sfm__estimator': Ridg...	0.792715	KNeighborsClassifier(algorithm='auto', leaf_si...	0.816383	0.849072	SFM
0	{'clf__n_neighbors': 9, 'pca__n_components': 5}	0.801261	KNeighborsClassifier(algorithm='auto', leaf_si...	0.815620	0.843159	PCA
5	{'clf__n_neighbors': 7, 'pca__n_components': 5}	0.797113	KNeighborsClassifier(algorithm='auto', leaf_si...	0.814733	0.855945	PCA

Figure 12:



	best_params	best_score	estimator	test_score	train_score	Method
0	{'clf__C': 10.0, 'pca__n_components': 5}	0.809229	SVC(C=10.0, cache_size=200, class_weight=None,...	0.874728	0.988817	PCA
29	{'clf__C': 10.0, 'feature_selection__k': 15}	0.812500	SVC(C=10.0, cache_size=200, class_weight=None,...	0.863636	0.940341	SKB
1	{'clf__max_features': 'auto', 'clf__n_estimato...	0.827744	(DecisionTreeClassifier(class_weight=None, cri...	0.849294	1.000000	PCA
28	{'clf__max_features': 'auto', 'clf__n_estimato...	0.801136	(DecisionTreeClassifier(class_weight=None, cri...	0.840909	1.000000	SKB
30	{'clf__C': 10.0, 'sfm__estimator': RidgeClassi...	0.781250	SVC(C=10.0, cache_size=200, class_weight=None,...	0.840909	0.889205	SFM

Figure 13:

- n_components = 5

While these results aren't very high, they are higher than the benchmark results. Regularization and feature selections/reduction improved the test score by 12% for the UCI dataset and 20% for the Database dataset. This shows that by narrowing down the datasets to the top 20 salient features and using other regularization methods in the pipelines, the new models are able to better classify the the target.