models

October 28, 2018

— title: Technical test for Junior Data Scientist position at Datrik Intelligence author: Antonio Ortega date: October 28, 2018—

1 Load libraries

```
In [25]: import pandas as pd
         import matplotlib.pyplot as plt
         import numpy as np
         import os.path
         from sklearn.model_selection import GridSearchCV
         from sklearn.decomposition import PCA
         from sklearn.model_selection import train_test_split
         from sklearn.linear_model import Ridge
         from sklearn.ensemble import GradientBoostingClassifier
         from sklearn.ensemble import AdaBoostClassifier
         from sklearn.ensemble import RandomForestClassifier
         from sklearn.metrics import accuracy_score
         from sklearn.metrics import roc_auc_score
         from sklearn.metrics import roc_curve
         from sklearn.metrics import auc
         from sklearn.preprocessing import StandardScaler
```

2 Load datasets

```
In [29]: x_train_mca = pd.read_csv(os.path.join("proc_data", "x_train_mca.csv"))
         x_train = pd.read_csv(os.path.join("proc_data", "x_train.csv"))
         x_train = pd.concat([x_train, x_train_mca.filter(like='MCA_')], axis=1)
         # x_test_mca = pd.read_csv(os.path.join("proc_data", "x_test_mca.csv"))
         y_train = pd.read_csv(os.path.join("proc_data", "y_train.csv"))
         X_train, X_test, y_train, y_test = train_test_split(x_train, y_train, test_size=0.33, random_s
```

```
In [30]: print("{} features loaded".format(X_train.columns.values.shape[0]))
         feature_count = np.unique([e.replace(".", "_").split("_")[0] for e in X_train.columns.values.t
         pd.DataFrame({"feature":feature_count[0], "count": feature_count[1]})
```

```
815 features loaded
```

```
Out[30]:    feature  count
         0      MCA      5
         1   binary      3
         2  counter      7
         3     edad      1
         4       et    719
         5  farmaco     21
         6  nominal     51
         7  ordinal      2
         8     raza      5
         9     sexo      1
```

```
In [4]: # plt.scatter(X_train["Dim 1"], X_train["Dim 2"], c=y_train.Y)
        # plt.show()
```

```
In [31]: scaler = StandardScaler()
         X_train_scaled = scaler.fit_transform(X_train)
         X_test_scaled = scaler.transform(X_test)
```

```
/home/antortjim/anaconda3/envs/ML/lib/python3.6/site-packages/sklearn/preprocessing/data.py:617: DataCon
  return self.partial_fit(X, y)
/home/antortjim/anaconda3/envs/ML/lib/python3.6/site-packages/sklearn/base.py:462: DataConversionWarning
  return self.fit(X, **fit_params).transform(X)
/home/antortjim/anaconda3/envs/ML/lib/python3.6/site-packages/ipykernel_launcher.py:3: DataConversionWar
  This is separate from the ipykernel package so we can avoid doing imports until
```

```
In [32]: # pca = PCA(n_components=2, svd_solver=’full’)
         # pca.fit(X_train_scaled)
         # X_train_transform = pd.DataFrame(data=pca.transform(X_train_scaled), columns=["PC1","PC2"])
         # X_train_transform.head()
```

```
Out[32]:         PC1       PC2
         0 -1.035688 -1.141891
         1 -1.088252  1.600154
         2  1.883554  2.612355
         3 -0.372338 -1.955333
         4 -0.039655  3.380544
```

```
In [37]: ridge = Ridge(normalize=True, max_iter=2000)
         parameters = {"alpha": [0, 1, 5, 7, 9, 10]}
         ridge_cv = GridSearchCV(estimator=ridge, param_grid=parameters, cv=5)
         ridge_cv.fit(X_train, y_train)
```

```
Out[37]: GridSearchCV(cv=5, error_score=’raise-deprecating’,
                estimator=Ridge(alpha=1.0, copy_X=True, fit_intercept=True, max_iter=2000,
            normalize=True, random_state=None, solver=’auto’, tol=0.001),
                fit_params=None, iid=’warn’, n_jobs=None,
                param_grid={’alpha’: [0, 1, 5, 7, 9, 10]}, pre_dispatch=’2*n_jobs’,
                refit=True, return_train_score=’warn’, scoring=None, verbose=0)
```

```
In [38]: y_predict=np.round(ridge_cv.predict(X_train))
```

```
In [39]: ridge_cv.cv_results_["params"]
         ridge_cv.best_estimator_
```

```
Out[39]: Ridge(alpha=1, copy_X=True, fit_intercept=True, max_iter=2000, normalize=True,
            random_state=None, solver=’auto’, tol=0.001)
```

```
In [40]: ridge_cv.score(X_train, y_train)
```

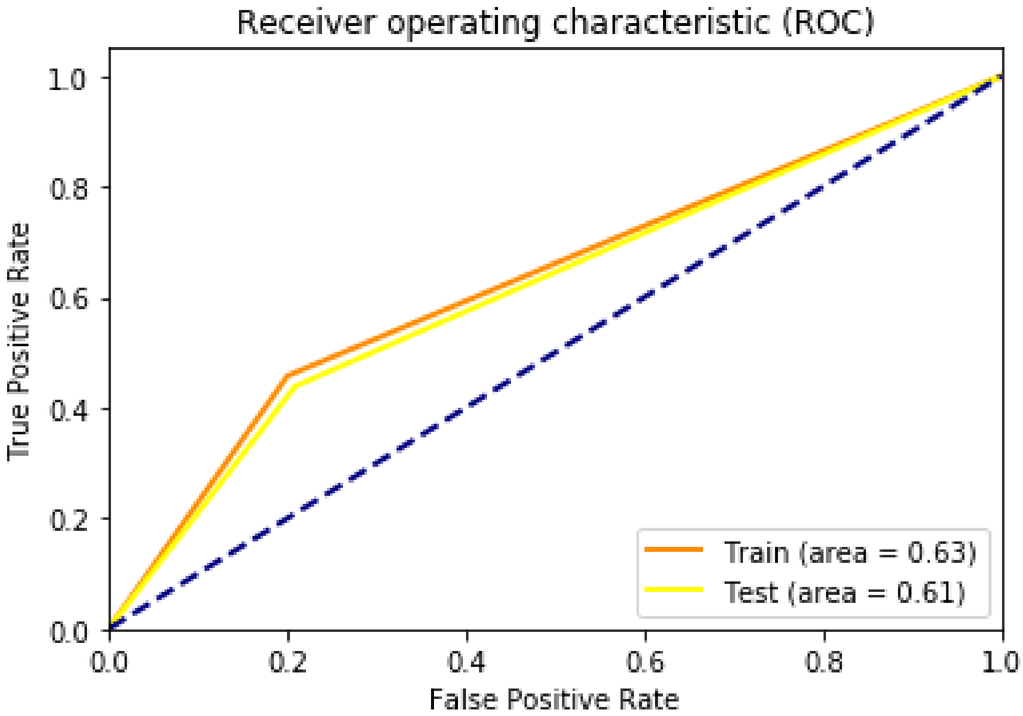
```
Out[40]: 0.10634049318208794
```

```
In [41]: y_predict_train = ridge_cv.predict(X_train)
         y_predict_train_bin = np.round(y_predict_train)
         print("Accuracy score (train): {0:.3f}".format(accuracy_score(y_train, y_predict_train_bin, no
         print("AUC score (train): {0:.3f}".format(roc_auc_score(y_train, y_predict_train_bin)))

         y_predict_test = ridge_cv.predict(X_test)
         y_predict_test_bin = np.round(y_predict_test)
         print("Accuracy score (test): {0:.3f}".format(accuracy_score(y_test, y_predict_test_bin, normal
         print("AUC score (test): {0:.3f}".format(roc_auc_score(y_test, y_predict_test_bin)))
```

```
Accuracy score (train): 0.643
AUC score (train): 0.629
Accuracy score (test): 0.627
AUC score (test): 0.615
```

```
In [42]: def plot_roc(y_train, y_predict_train_bin, y_test, y_predict_test_bin):
           lw=2
           fpr = dict()
           tpr = dict()
           roc_auc = dict()
           fpr[0], tpr[0], _ = roc_curve(y_true=y_train, y_score = y_predict_train_bin)
           roc_auc[0] = auc(fpr[0], tpr[0])
           fpr[1], tpr[1], _ = roc_curve(y_true=y_test, y_score = y_predict_test_bin)
           roc_auc[1] = auc(fpr[1], tpr[1])
           cols = ['darkorange', "yellow"]
           datasets = ["Train", "Test"]
           plt.figure()
           for i in range(2):
             plt.plot(fpr[i], tpr[i], color=cols[i],
                      lw=lw, label='%s (area = %0.2f)' % (datasets[i], roc_auc[i]))

           plt.plot([0, 1], [0, 1], color='navy', lw=lw, linestyle='--')
           plt.xlim([0.0, 1.0])
           plt.ylim([0.0, 1.05])
           plt.xlabel('False Positive Rate')
           plt.ylabel('True Positive Rate')
           plt.title('Receiver operating characteristic (ROC)')
           plt.legend(loc="lower right")
           plt.show()
```

```
In [43]: plot_roc(y_train, y_predict_train_bin, y_test, y_predict_test_bin)
```



```
In [44]: learning_rates = [0.05, 0.1, 0.3, 0.5]
         gb = GradientBoostingClassifier(random_state = 0, subsample = 0.9, n_estimators=500)
         param_grid = dict(learning_rate = learning_rates,
                           #n_estimators = [10,50,100],
                           max_features=[1,5,10], max_leaf_nodes = [2,3,4])
         gb_cv = GridSearchCV(estimator=gb, param_grid=param_grid, cv=5, verbose=True,n_jobs=2)
         gb_cv.fit(X_train_scaled, y_train.values.reshape((-1,)))
```

```
Fitting 5 folds for each of 36 candidates, totalling 180 fits

[Parallel(n_jobs=2)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=2)]: Done  46 tasks      | elapsed:  5.4min
[Parallel(n_jobs=2)]: Done 180 out of 180 | elapsed: 20.6min finished
```

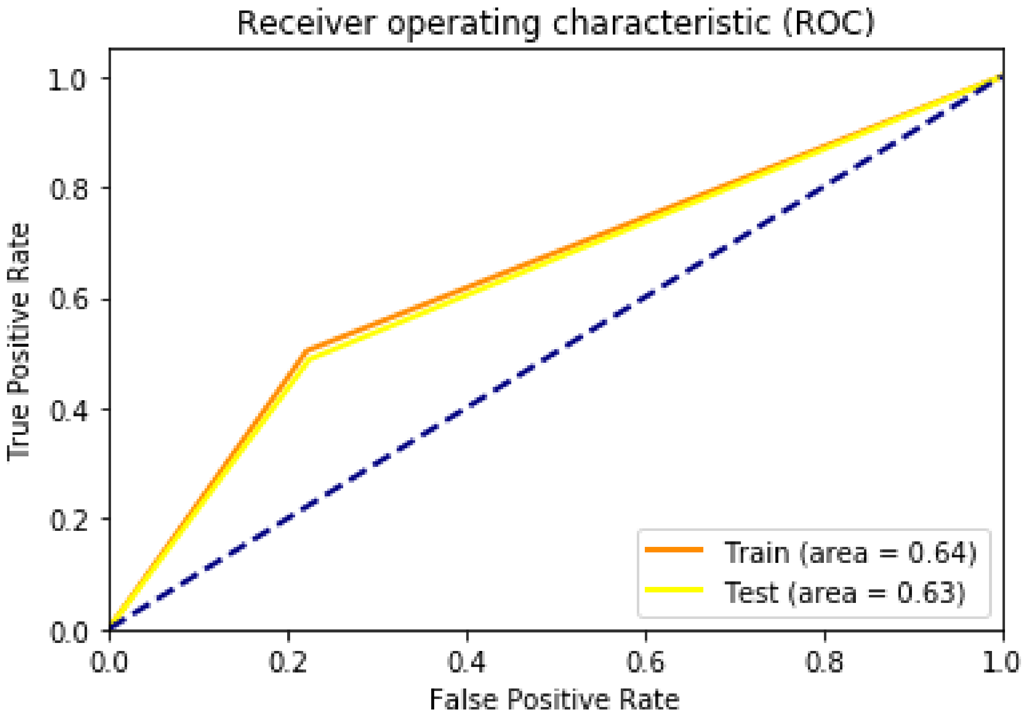
```
Out[44]: GridSearchCV(cv=5, error_score='raise-deprecating',
                estimator=GradientBoostingClassifier(criterion='friedman_mse', init=None,
                      learning_rate=0.1, loss='deviance', max_depth=3,
                      max_features=None, max_leaf_nodes=None,
                      min_impurity_decrease=0.0, min_impurity_split=None,
                      min_samples_leaf=1, min_sampl...      subsample=0.9, tol=0.0001, validation_fract
                      verbose=0, warm_start=False),
                fit_params=None, iid='warn', n_jobs=2,
                param_grid={'learning_rate': [0.05, 0.1, 0.3, 0.5], 'max_features': [1, 5, 10], 'max_leaf
                pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
                scoring=None, verbose=True)
```

```
In [46]: y_predict_train = gb_cv.predict(X_train_scaled)
         y_predict_train_bin = np.round(y_predict_train)
```

```
print("Accuracy score (training): {0:.3f}".format(accuracy_score(y_train, y_predict_train_bin,
print("AUC score (training): {0:.3f}".format(roc_auc_score(y_train, y_predict_train_bin)))

y_predict_test = gb_cv.predict(X_test_scaled)
y_predict_test_bin = np.round(y_predict_test)
print("Accuracy score (test): {0:.3f}".format(accuracy_score(y_test, y_predict_test_bin, normal
print("AUC score (test): {0:.3f}".format(roc_auc_score(y_test, y_predict_test_bin)))
# print("Learning rate: ", learning_rate)
# print("Accuracy score (training): {0:.3f}".format(gb.score(X_train_scaled, y_train)))
# print("Accuracy score (validation): {0:.3f}".format(gb.score(X_test_scaled, y_test)))
```

```
Accuracy score (training): 0.652
AUC score (training): 0.641
Accuracy score (test): 0.642
AUC score (test): 0.632
```

```
In [48]: plot_roc(y_train, y_predict_train_bin, y_test, y_predict_test_bin)
```


Receiver operating characteristic (ROC)
True Positive Rate
False Positive Rate
Train (area = 0.64)
Test (area = 0.63)