Challenge mdi 343

Machine Learning - MS BigData

Christophe Thibault

Telecom ParisTech

February 11, 2018

Table of contents

Challenge nmdi 343

Find a weight matrix $\mathscr{M}$ 15x15 that represents all combinations between 14 algorithms with time constraint (600ms) with $p = 14$ and $n = 2048840$:

$$\begin{pmatrix} 0 & \cdots & w_{1p} \\ \vdots & \ddots & \vdots \\ 0 & \cdots & w_{np} \end{pmatrix}$$

```python
# Running time of each algorithm (in milliseconds)
alg_times = np.zeros((14,1))
alg_times[0] = 163
alg_times[1] = 163
alg_times[2] = 190
alg_times[3] = 190
alg_times[4] = 206
alg_times[5] = 206
alg_times[6] = 120
alg_times[7] = 120
alg_times[8] = 83
alg_times[9] = 83
alg_times[10] = 83
alg_times[11] = 83
alg_times[12] = 170
alg_times[13] = 170
```

Figure 1: Algorithms running times

Data exploration

In [4]:
```python
import random
import matplotlib.pyplot as plt
import seaborn as sns
f, ax = plt.subplots(figsize=(10, 8))
corr = train.corr()
sns.heatmap(corr, mask=np.zeros_like(corr, dtype=np.bool), cmap=sns.diverging_palette(220, 10, as_cmap=True),
            square=True, ax=ax)
```

Out[4]: `<matplotlib.axes._subplots.AxesSubplot at 0x1154747b8>`

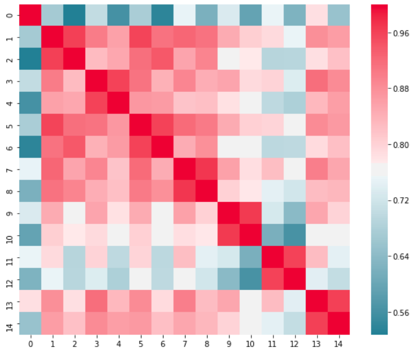

Figure 2: Matrix correlation

Data exploration

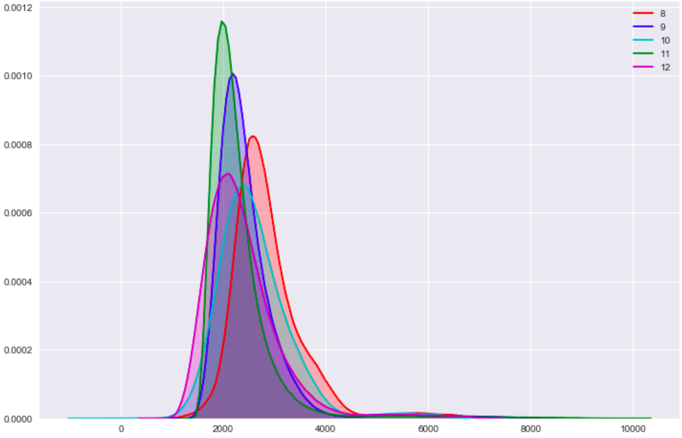

Figure 3: Probability density of scores (algorithms 8, 9, 10, 11 and 12)

Recurcive Feature Elimination

Recurcive Feature Elimination (REF)

In [12]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.feature_selection import RFE
rfc = RandomForestClassifier()

# create the RFE model and select 4 attributes
rfe_model = RFE(rfc, 5, step=1)
rfe_model = rfe_model.fit(X, y)

# summarize the selection of the attributes
print(rfe_model.support_)
print(rfe_model.ranking_)
```

```
[False False False False False False  True  True  True  True False False
 False  True]
[ 5 10  3  8  7  9  1  1  1  1  2  6  4  1]
```

In [13]:
```
# visualize the selected features:
mask = rfe_model.get_support()
plt.matshow(mask.reshape(1, -1), cmap='gray_r')
plt.xlabel("Sample index");
```

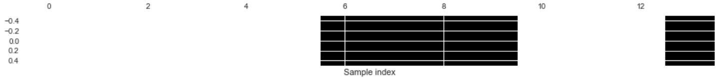

Figure 4: Recurcive Feature Elimination with random forest - Five most important are 7, 8, 9, 10 and 14

Density probability - Random Forest on the whole data set

Best Features importance

with 8 - 9 - 10 - 11 - 13 - 14 algorithms

```
In [14]: sns.set(rc={"figure.figsize": (12, 8)})
         ax = sns.kdeplot(train[7], shade=True, color="r")
         sns.kdeplot(train[8], ax=ax, shade=True, color="b")
         sns.kdeplot(train[9], ax=ax, shade=True, color="c")
         sns.kdeplot(train[10], ax=ax, shade=True, color="g")
         sns.kdeplot(train[14], ax=ax, shade=True, color="m")

Out[14]: <matplotlib.axes._subplots.AxesSubplot at 0x10d65f748>
```

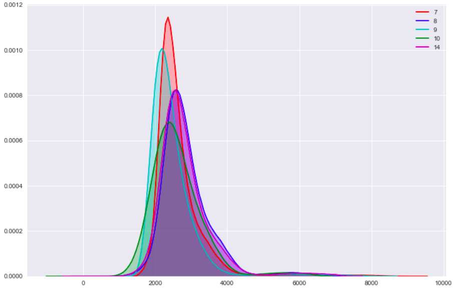

Figure 5: Density probability of scores (algorithms 7, 8, 9, 10 and 14

Find the combination : Use of Feature Importances

Best score

with **8** - **9** - **10** - **11** - **14** algorithms

3) PolynomialFeatures sur 8-9-10-11-14 avec AdaBoost

```
X_red = np.array([X[:,7],X[:,8],X[:,9], X[:,10], X[:,13]]).reshape(2048840,5)
from sklearn.preprocessing import PolynomialFeatures
model = PolynomialFeatures(degree=2)
X_poly = model.fit_transform(X_red)
from sklearn.ensemble import AdaBoostClassifier
ada = AdaBoostClassifier(n_estimators=50)
ada.fit(X_poly, y)
print(ada.feature_importances_)
```

```
[ 0.    0.1   0.02  0.04  0.04  0.06  0.04  0.04  0.06  0.04  0.08  0.06
  0.04  0.1   0.    0.06  0.02  0.04  0.08  0.02  0.06]
```

Le score sur le test est de 0.065, ce qui est mieux. Essayons maintenant avec une simple combinaison linéaire.

3b) Adaboost sur la combinaison 8, 9, 10, 11, et 14

```
X_red = np.array([X[:,7],X[:,8],X[:,9], X[:,10], X[:,13]]).reshape(2048840,5)
from sklearn.ensemble import AdaBoostClassifier
ada = AdaBoostClassifier(n_estimators=50)
ada.fit(X_red, y)
print(ada.feature_importances_)
```

```
[ 0.18  0.2   0.18  0.24  0.2 ]
```

Figure 6: Adaboost (with estimators=50) on linear and quadradic (degree=2) combination

FRR minimize

- Find the best combination that minimizes the FRR (with time constraint)
- Use of itertools to compute all combinations and calcule the FRR for each one

Results

Best combination : 8, 9, 10, 11, and 14

- Use of AdaBoost classifier $\Longrightarrow$ find the coefficients of each algorithm
- M[0,8]=0.18; M[0,9]=0.2; M[0,10]=0.18; M[0,11]=0.14; M[0,14]=0.2

Conclusion

- Best score **0.0638** with 8-9-10-11-14 algorithms
- linear combination M[0, 8] = 0.201 ; M[0,9] = 0.263; M[0, 10] = 0.181 ; M[0,11] = 0.131 ; M[0;14] = 0.147
- In any cases, best score with **linear combination** rather than quadratic combination.