Trabalho Final em Laboratório Virtual

Disciplina	RDP – Reconhecimento de Padrões

Objetivos

Estruturar um sistema de reconhecimento de padrões para identificar risco de inadimplência. Nessa atividade desenvolveremos:

- ✓ Capacidade de estruturar um sistema de reconhecimento para a classificação binária
- ✓ Aplicar metodologias de redução de dimensionalidade
- ✓ Avaliar comparativamente o desempenho diferentes algoritmos de classificação

Base de dados

Utilizaremos para essa tarefa a base de dados Default of Credit Cards Dataset[1], que é composta possui 25 variáveis e 1000 amostras disponível no arquivo “lv-credit-card-default-1000.csv”. O campo DEFAULTED possui um valor binário o qual o valor 1 significa inadimplente.

Atividades

O trabalho deve obrigatoriamente conter:

- ✓ Mínimo de dois algoritmos diferentes para a classificação
- ✓ Aplicação de validação cruzada (múltiplos splits é opcional) para o treinamento
- ✓ Aplicação de duas estratégias (algoritmos) para a redução da dimensionalidade
- ✓ Avaliação de, no mínimo, duas métricas diferentes (dê preferência por AUC)
- ✓ Medição do tempo de execução e treinamento de cada um deles
- ✓ Extra: escolha de um Ensemble (ex.: Random Forest) como um dos métodos
- ✓ Criação de tabela com os resultados obtidos por cada um deles em termos das métricas e dos tempos obtidos

[1] Dados abertos disponíveis em: https://www.kaggle.com/uciml/default-of-credit-card-clients-dataset

Referências

- ✓ Implementação de uma árvore de decisão:
  http://scikit-learn.org/stable/modules/tree.html

```
>>> from sklearn.datasets import load_iris
>>> from sklearn import tree
>>> iris = load_iris()
>>> clf = tree.DecisionTreeClassifier()
>>> clf = clf.fit(iris.data, iris.target)
```

- ✓ Implementação de uma SVM:
  http://scikit-learn.org/stable/modules/svm.html

```
>>> from sklearn import svm
>>> X = [[0, 0], [1, 1]]
>>> y = [0, 1]
>>> clf = svm.SVC()
>>> clf.fit(X, y)
```

- ✓ Implementação de uma Random Forest e métodos Ensemble
  http://scikit-learn.org/stable/modules/ensemble.html

```
>>> from sklearn.ensemble import RandomForestClassifier
>>> X = [[0, 0], [1, 1]]
>>> Y = [0, 1]
>>> clf = RandomForestClassifier(n_estimators=10)
>>> clf = clf.fit(X, Y)
```

```
>>> from sklearn.ensemble import BaggingClassifier
>>> from sklearn.neighbors import KNeighborsClassifier
>>> bagging = BaggingClassifier(KNeighborsClassifier(),
...                             max_samples=0.5, max_features=0.5)
```

- ✓ Implementação do PCA com o Spark:

  http://scikit-learn.org/stable/auto_examples/decomposition/plot_pca_iris.html

```
>>> plt.cla()
>>> pca = decomposition.PCA(n_components=3)
>>> pca.fit(X)
>>> X = pca.transform(X)
```