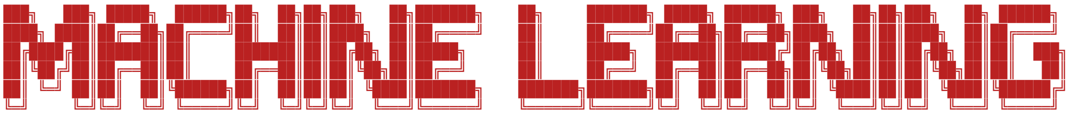
```python
# Portfolio

# Machine Learning / Deep Learning

# Author: Leonardo Vinícius Damasio da Silva | Cientista de Dados
# Date: 07/10/2019


print("""\


```



```python


                                                                                                        by Leonardo Damasio
""")


# Tools

print("\nImporting tools")

# Basic Tools
import pandas as pd # Pandas
import numpy as np # NumPy
import matplotlib.pyplot as plt # MatPlotLib
import operator # Operator
import os # Operating System
import pickle # Pickle
import webbrowser # Web Browser

# Data Wrangling Tools
from sklearn.model_selection import train_test_split # Train / Test Split
from sklearn.preprocessing import LabelEncoder # Transforms into numerical (Only if you have categorical records)

# Results Tools
from sklearn.metrics import confusion_matrix, accuracy_score # Simple Confusion Matrix and Accuracy Score
from yellowbrick.classifier import ConfusionMatrix # Confusion Matrix Plot
from matplotlib.pylab import rcParams # Plot Size

# Extra Tools
from sklearn.ensemble import ExtraTreesClassifier # Variable Importances

print("Successfully imported tools")


# Results Folder

print("\nCreating the results folder")

try:
    os.mkdir("results")
    print ("Successfully created the directory <results>.")
except OSError:
    print ("Directory <results> already exists.")


# Classifying Algorithms (Choose One)

random = 32475
possible = range(7)
choice = ""

while choice not in possible:

    try:

        choice = int(input("""
Choose the number of your algorithm:

0 for Naive Bayes Classifier
1 for K-Nearest Neighbors Classifier
2 for Decision Tree Classifier
3 for Support Vector Machine Classifier
4 for Random Forest Classifier
5 for Extreme Gradient Boosted Trees Classifier
6 for Deep Learning Multilayer Perceptron Neural Networks

Choice: """))

    except:
        pass

    if choice not in possible:
        print("You must choose one of these numbers: ", list(possible))

if choice == 0: print("Naive Bayes Classifier\n")
if choice == 1: print("K-Nearest Neighbors Classifier\n")
if choice == 2: print("Decision Tree Classifier\n")
if choice == 3: print("Support Vector Machine Classifier\n")
if choice == 4: print("Random Forest Classifier\n")
if choice == 5: print("Extreme Gradient Boosted Trees Classifier\n")
if choice == 6: print("Deep Learning Multilayer Perceptron Neural Networks\n")



# Function: Training Model

def train_model():


    # Model Instance

    if choice == 0:
        from sklearn.naive_bayes import GaussianNB
        model = GaussianNB()

```

```
    elif choice == 1:
        from sklearn.neighbors import KNeighborsClassifier
        model = KNeighborsClassifier(n_neighbors = 3)

    elif choice == 2:
        from sklearn.tree import DecisionTreeClassifier
        model = DecisionTreeClassifier(random_state=random)
        import graphviz # Graph visualization
        from sklearn.tree import export_graphviz # Creates a .dot for a Decision Tree Visualization. To visualize, copy and paste the content inside http://www.we

    elif choice == 3:
        from sklearn.svm import SVC
        model = SVC(probability=True, random_state=random)

    elif choice == 4:
        from sklearn.ensemble import RandomForestClassifier
        model = RandomForestClassifier(n_estimators = 1000, random_state=random)

    elif choice == 5:
        from xgboost import XGBClassifier
        model = XGBClassifier(base_score=0.5,
                              booster="gbtree",
                              colsample_bylevel=1,
                              colsample_bynode=1,
                              colsample_bytree=0.8,
                              gamma=0,
                              learning_rate=0.2,
                              max_delta_step=0,
                              max_depth=5,
                              min_child_weight=1,
                              missing=None,
                              n_estimators=1000,
                              n_jobs=1,
                              nthread=None,
                              objective="binary:logistic",
                              random_state=random,
                              reg_alpha=0,
                              reg_lambda=1,
                              scale_pos_weight=1,
                              seed=None,
                              silent=None,
                              subsample=0.8,
                              verbosity=1)

    elif choice == 6:
        from keras.models import Sequential
        from keras.layers import Dense, Dropout
        from keras.utils import np_utils
        model = Sequential()
        model.add(Dense(units=156, activation="relu", input_dim=20))
        model.add(Dropout(0.2))
        model.add(Dense(units=100, activation="relu"))
        model.add(Dense(units=80, activation="relu"))
        model.add(Dense(units=60, activation="relu"))
        model.add(Dense(units=40, activation="relu"))
        model.add(Dense(units=20, activation="relu"))
        model.add(Dense(units=2, activation="softmax"))
        model.summary()
        model.compile(optimizer="adam", loss="categorical_crossentropy", metrics=["accuracy"])

    print("Success")


    # Datasets Import

    print("\nImporting dataset")

    dataset = pd.read_csv("credit.csv", sep = ",")
    print("Dataset imported")
    print("Dataset Shape:", dataset.shape)


    # X / Y Split

    x = dataset.iloc[:,0:20]
    labels = [i for i in x.columns]

    print("\n***** X *****\n")
    print(pd.DataFrame(x).head())

    x = x.values
    labelencoder = LabelEncoder()
    transform = [0,2,3,5,6,8,9,11,13,14,16,18,19]

    for i in transform:
        x[:,i] = labelencoder.fit_transform(x[:,i])

    x = pd.DataFrame(x)
    x.columns = labels
    print("\nX categorical to numerical\n")
    print(pd.DataFrame(x).head())


    y = dataset.iloc[:,20]
    print("\n***** Y *****\n")
    print(pd.DataFrame(y).head())

    y = y.values
    y = labelencoder.fit_transform(y)
    y = pd.DataFrame(y)
    y.columns = ["class"]
    print("\nY categorical to numerical\n")
    print(pd.DataFrame(y).head())


    # Train / Test Split

    x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=random)

    if choice == 6: # Neural Network
        y_test = np_utils.to_categorical(y_test)
        y_train = np_utils.to_categorical(y_train)
```

```
224
225     # Displaying Train / Test Proportions
226
227     print("\nTrain / Test Split\n")
228
229     print("X Train:")
230     print(x_train.shape[0], "records")
231     print(x_train.shape[1], "predictor/explanatory/independent variables\n")
232
233     print("Y Train:")
234     print(y_train.shape[0], "records")
235     try:
236         print(y_train.shape[1], "predicted/response/dependent variables\n")
237     except:
238         print("1 predicted/response/dependent variable\n")
239
240     print("X Test:")
241     print(x_test.shape[0], "records")
242     print(x_test.shape[1], "predictor/explanatory/independent variables\n")
243
244     print("Y Test:")
245     print(y_test.shape[0], "records")
246     try:
247         print(y_test.shape[1], "predicted/response/dependent variables\n")
248     except:
249         print("1 predicted/response/dependent variable\n")
250
251
252     # Model Fit
253
254     if choice == 5: # XGBClassifier
255         x_train = x_train.values
256         y_train = y_train.values
257         x_test = x_test.values
258         y_test = y_test.values
259
260     if choice != 6: # Not a Neural Network
261         model.fit(x_train, y_train)
262         print("\nModel Fitted")
263
264     if choice == 6: # Neural Network
265         history = model.fit(x_train, y_train, epochs=12, validation_data=(x_test, y_test))
266         print("\nModel Fitted")
267
268         history.history.keys()
269         rcParams["figure.figsize"] = 20, 6
270         plt.figure(1)
271         plt.subplot(1,2,1).set_title("\nval_loss\n", fontsize=20)
272         plt.plot(history.history["val_loss"])
273         plt.xlabel("\nIterations\n", fontsize=15)
274         plt.ylabel("\nVal Loss\n", fontsize=15)
275         plt.subplot(1,2,2).set_title("\nval_acc\n", fontsize=20)
276         plt.plot(history.history["val_acc"])
277         plt.xlabel("\nIterations\n", fontsize=15)
278         plt.ylabel("\nVal Acc\n", fontsize=15)
279         plt.grid(alpha=0.5)
280         plt.yticks(fontsize=12)
281         plt.tight_layout()
282         plt.show()
283
284     if choice == 2: # Decision Tree Classifier
285         export_graphviz(model, out_file = "tree.dot")
286         print("<tree.dot> file exported")
287
288     pickle.dump(model, open("results/model"+str(choice)+".sav", "wb"))
289
290
291     # Test Prediction
292
293     pred_y_test = model.predict(x_test)
294     print("\nTest Predicted")
295     print("\nPredictions\n")
296
297     if choice != 6: # Not a Neural Network
298         print(pd.DataFrame(pred_y_test).head())
299
300     if choice == 6: # Neural Network
301         y_test_un = [np.argmax(i) for i in y_test]
302         pred_y_test_un = [np.argmax(i) for i in pred_y_test]
303         print(pred_y_test_un[0:30])
304
305
306     # Test Prediction Probabilities
307
308     if choice != 6: # Not a Neural Network
309         probability = model.predict_proba(x_test)
310
311     if choice == 6: # Neural Network
312         probability = pred_y_test
313
314     print("\nTest Prediction Probabilities\n")
315     print(pd.DataFrame(probability.round(4)).head())
316
317
318     # Accuracy Score
319
320     if choice != 6: # Not a Neural Network
321         score = accuracy_score(y_test, pred_y_test)
322
323     if choice == 6: # Neural Network
324         score = accuracy_score(y_test_un, pred_y_test_un)
325
326     print("\nAccuracy Score: ", score)
327
328
329     # Simple Confusion Matrix
330
331     if choice != 6: # Not a Neural Network
332         matrix = confusion_matrix(y_test, pred_y_test)
333
334     if choice == 6: # Neural Network
335         matrix = confusion_matrix(y_test_un, pred_y_test_un)
```

```
    print("\nMatrix\n")
    print(pd.DataFrame(matrix))


    # Confusion Matrix PLot

    if choice != 6: # Not a Neural Network
        matrix_plot = ConfusionMatrix(model)
        rcParams["figure.figsize"] = 5, 5
        matrix_plot.fit(x_train, y_train)
        matrix_plot.score(x_test, y_test)
        matrix_plot.poof(outpath="results/matrix.png", dpi=300) # Only if you want to save the plot as an image
        matrix_plot.poof()

        print("\nImage <matrix.png> saved.\n")

    if choice == 6: # Neural Network
        pass


    # Variables Importances

    forest = ExtraTreesClassifier(n_estimators=1000, random_state=random)
    forest.fit(x_train, y_train)
    importances = forest.feature_importances_

    dic = dict(zip(labels, importances.round(4)))
    sort_values = sorted(dic.items(), key=operator.itemgetter(1), reverse=False)
    sorted_importances = pd.DataFrame(sort_values)

    print("\nVariables Importances\n")
    print(pd.DataFrame(sorted_importances.values, columns=["Variable", "Importance"]))


    # Variables Importances PLot

    plt.rcParams["figure.figsize"] = 12, 10
    plt.scatter(sorted_importances[1], sorted_importances[0])
    plt.title("\nImportances\n", fontsize=20)
    plt.xlabel("\nImportance (0~1)\n", fontsize=15)
    plt.ylabel("\nVariable\n", fontsize=15)
    plt.grid(alpha=0.5)
    plt.yticks(fontsize=13)
    plt.tight_layout()
    plt.savefig("results/importances.png", format="png", dpi = 300, bbox_inches="tight") # Only if you want to save the plot as an image
    plt.show()

    print("\nImage <importances.png> saved.\n")


    # Exporting Importances

    lista = []

    index = 0
    for i in labels:
        lista.append(str(round(importances[index]*100,2)) + "% | " + str(i))
        index += 1

    file = open("results/importances.csv", "w")

    file.write("Importance|Variable\n")

    index = 0
    while index < len(labels):
        file.write(str(lista[index])+"\n")
        index += 1

    file.close()

    print("\nFile <importances.csv> saved.\n")


    # Exporting Predictions

    file = open("results/predictions.csv", "w")

    file.write("Key|x_test|y_test|pred_y_test|Probability\n")

    index = 0

    while index < len(pred_y_test):

        if choice != 6: # Not a Neural Network
            file.write(str(index) + "|" + str(np.array(x_test)[index]) + "|" + str(np.array(y_test)[index]) + "|" + str(np.array(pred_y_test)[index]) + "|" + str(

        if choice == 6: # Neural Network
            file.write(str(index) + "|" + str(np.array(x_test)[index]) + "|" + str(np.array(y_test_un)[index]) + "|" + str(np.array(pred_y_test_un)[index]) + "|"

        index += 1

    file.close()

    print("\nFile <predictions.csv> saved.\n")


# Function: Predict new data

def predict_new():


    # Importing New Dataset

    model = pickle.load(open("results/model"+str(choice)+".sav", "rb"))
    new_dataset = pd.read_csv("novocredit.csv", sep = ",")


    # Defining X

    new_x = new_dataset
    labels = [i for i in new_x.columns]
```

```
448
449     print("\n***** NEW X *****\n")
450     print(pd.DataFrame(new_x).head())
451
452     new_x = new_x.values
453     labelencoder = LabelEncoder()
454     transform = [0,2,3,5,6,8,9,11,13,14,16,18,19]
455
456     for i in transform:
457         new_x[:,i] = labelencoder.fit_transform(new_x[:,i])
458
459     new_x = pd.DataFrame(new_x)
460     new_x.columns = labels
461     print("\nNEW X categorical to numerical\n")
462     print(pd.DataFrame(new_x).head())
463
464
465     # New Prediction
466
467     if choice == 5: # XGBClassifier
468         new_x = new_x.values
469
470     pred_new_y = model.predict(new_x)
471     print("\nPredicted")
472     print("\nPredictions\n")
473
474     if choice != 6: # Not a Neural Network
475         print(pd.DataFrame(pred_new_y).head())
476
477     if choice == 6: # Neural Network
478         pred_new_y_un = [np.argmax(i) for i in pred_new_y]
479         print(pred_new_y_un[0:30])
480
481
482     # New Prediction Probabilities
483
484     if choice != 6: # Not a Neural Network
485         new_probability = model.predict_proba(new_x)
486
487     if choice == 6: # Neural Network
488         new_probability = pred_new_y
489
490     print("\nNew Prediction Probabilities\n")
491     print(pd.DataFrame(new_probability.round(4)).head())
492
493
494     # Exporting Predictions
495
496     nome_arquivo = str(input("\nDigite o nome do arquivo onde serão salvas as predições: "))+".csv"
497
498     file = open("results/"+str(nome_arquivo), "w")
499
500     file.write("Key|new_x|pred_new_y_un|Probability\n")
501
502     index = 0
503
504     while index < len(pred_new_y):
505
506         if choice != 6: # Not a Neural Network
507             file.write(str(index) + "|" + str(np.array(new_x)[index]) + "|" + str(np.array(pred_new_y)[index]) + "|" + str(round(new_probability[index][1], 4)).re
508
509         if choice == 6: # Neural Network
510             file.write(str(index) + "|" + str(np.array(new_x)[index]) + "|" + str(np.array(pred_new_y_un)[index]) + "|" + str(round(new_probability[index][1], 4))
511
512         index += 1
513
514     file.close()
515
516     print("\nFile <"+str(nome_arquivo)+"> saved.\n")
517
518     input("\nPress ENTER to exit")
519
520
521 # Running Functions
522
523 try:
524     predict_new()
525 except:
526     train_model()
527     predict_new()
528
529
530 # *Leonardo Damasio* | **Data Scientist**
531
532 # LinkedIn
533 # www.linkedin.com/in/leonardodamasio
534 webbrowser.open("https://www.linkedin.com/in/leonardodamasio")
535
536 # GitHub
537 # www.github.com/leonardodamasio/
538 webbrowser.open("https://github.com/leonardodamasio")
539
540 # Email
541 # leoleonardo1996@hotmail.com
```