Projet n°8 :
“ Credit Scoring ”

Soutenance de Jury - Projet n°8 - Juin 2022

Ali Naama

Sommaire

I. Description du contexte et exploration du jeu de données

Description du contexte

- Une banque en ligne souhaiterait **adapter sa gestion du Credit Scoring en conséquence de l'acquisition d'une nouvelle filiale allemande**.

- Pour cela elle dispose du fichier client de la nouvelle filiale et demande aux équipes IT et Data Scientist de proposer un scoring qui soit accessible sur une plateforme Web et qui offre d'excellentes performance.

Objectifs :

En réponse à cette demande, je vais proposer une démarche projet adaptée au contexte client, en proposant un Credit Scoring au sein d'une plateforme cloud Big Data Amazon EMR.

Credit Scoring - Contexte

- Le credit scoring, ou encore scoring d'octroi, est un des outils mis en œuvre lors de l'analyse risque d'une demande de crédit par les prêteurs (Banque en ligne dans notre cas).

- Il est nécessaire de disposer :
  - d'un **produit** de crédit (ex : l'ouverture de crédit de 2.500 €)
  - **échantillon de clients** ayant eu accès à ce crédit
  - de **l'historique de remboursement** de ces clients

- Sur la base de ces informations, une **analyse des données personnelles** des clients sera réalisée pour permettre d'identifier les combinaisons de données qui sont les plus fréquentes lors de défaillances, et à partir de là, construire une grille de score qui permettra de prédire la probabilité de défaillance (versus de remboursement) de chacun des clients.

Jeu de données « Credi Scorong » adapté au contexte client

Informations sur les variables :

- ❑ Les données : sont extraites du site Download German Credit Dataset (german.csv) et correspondent à un contexte de données bancaires
- ❑ Le fichier résultant de ces opérations devient le fichier data_german.csv

Nombre de colonnes du fichier : 21

Nombre de lignes du fichier : 1000

Fichier	Nb lignes	Nb colonnes	Taux remplissage moyen	Doublons	Description
data_german.csv	1000	21	**100%**	N/A	

Valeurs clés du jeu de données

Champ	Valeur d'exemple	Type	Description
Status	A11, …A14	Catégorielle	Status of existing checking account, Ex : A11 : < 0
Duration		Numérique	Duration in month
History	A30, ..A34	Catégorielle	Credit history, Ex : A30 : no credits taken
Purpose	A40, A410	Catégorielle	Ex : A40 : car (new)
Amount		Numérique	Credit amount
Savings	A61, .,A64	Catégorielle	Savings account/bonds
Employment	A71, ..A75	Catégorielle	Present employment since
Rate		Numérique	Installment rate in percentage of disposable income
Situation	A91, ..A95	Catégorielle	Personal status and sex, Ex : A91 : male : divorced/separated
Debtors		Catégorielle	
Residence Since		Numérique	Present residence since
Property		Catégorielle	Ex : A121 : real estate
Age		Numérique	
Installment	A141, ..A143	Catégorielle	Other installment plans, Ex : A141 : bank, A142 : stores, A143 : none
Housing		Catégorielle	A151 : rent, A152 : own, A153 : for free
Number Credits		Numérique	Number of existing credits at this bank
Job	A171, A172, A173, A174	Catégorielle	Ex : A171 : unemployed/ unskilled - non-resident
People liable		Numérique	Number of people being liable to provide maintenance
Phone	A191, A192	Catégorielle	A191 : none, A192 : yes, registered under customer name
Foreign Worker		Catégorielle	A201 : yes, A202 : no
Score		Numérique	Ex : (1 = Good, 2 = Bad) : Variable à prédire

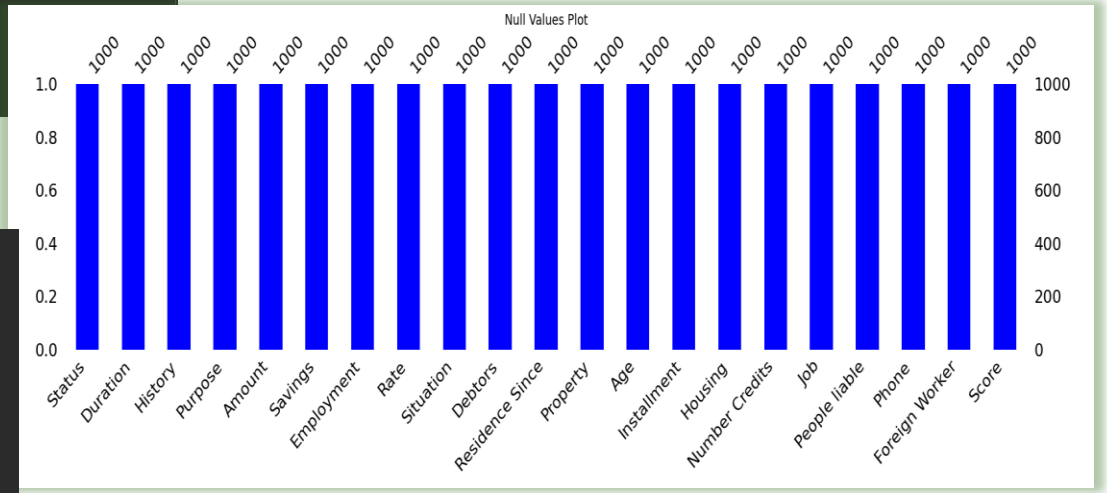
Nombre de colonnes du fichier : 21

Nombre de lignes du fichier : 1000

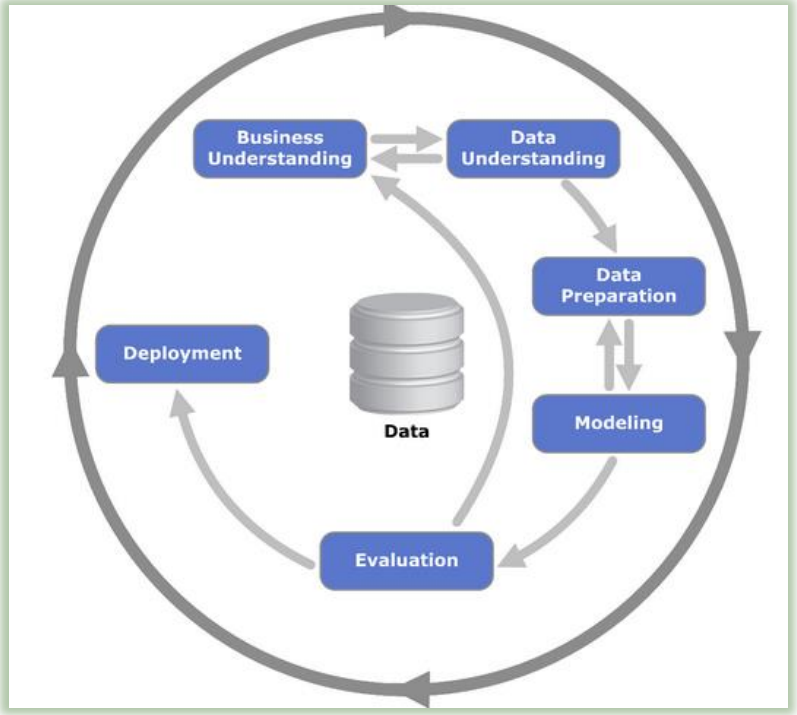
Credit Score – Taux de Remplissage par colonne

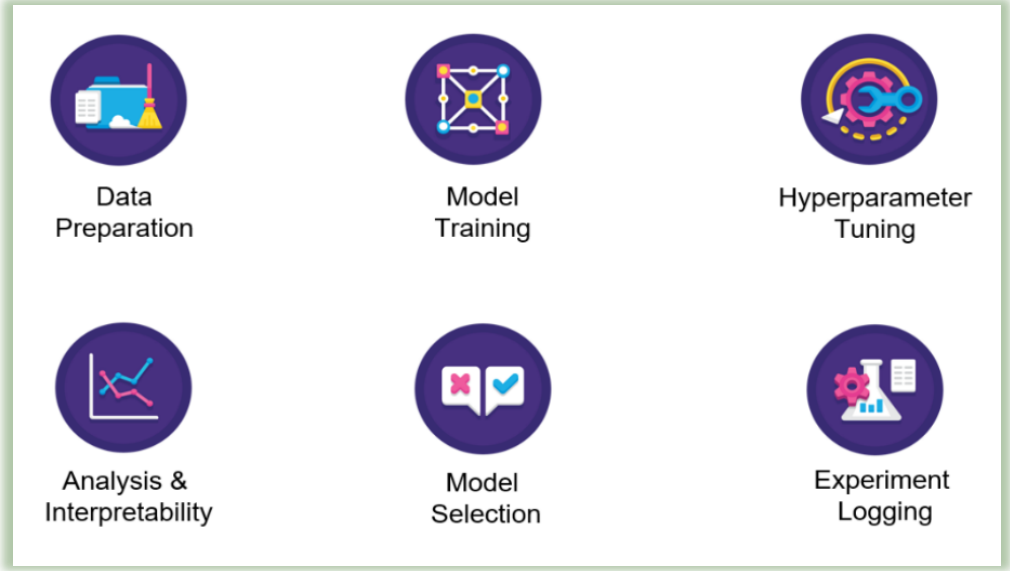

```
# Init Spark Session

spark = SparkSession \
    .builder \
    .appName("Credit_Score") \
    .getOrCreate()

# Read csv file
# By setting inferSchema=true , Spark will automatically go through the csv file and infer the
# schema of each column. This requires an extra pass over the file which will result in reading a file with inferSchema
# set to true being slower. But in return the dataframe will most likely have a correct schema given its input
df = spark.read.csv("data/data_german.csv", inferSchema=True, header=True)
# change value of score to get binary : 0 or 1
df = df.withColumn('Score', F.when(F.col("Score") == '2', 1).otherwise(0))
print(df.head())

pandasDF = df.toPandas()
print(pandasDF)
```

```
plt.title("Null Values Plot")
#ms.bar(df, color = 'slategrey')
ms.bar(pandasDF, color = 'blue')
plt.savefig( 'image/Chart Null Values Plot_' +  'data.png', bbox_inches='tight')
plt.close()
#plt.show()
```



Méthodologie de Data Science « CRISP DM »

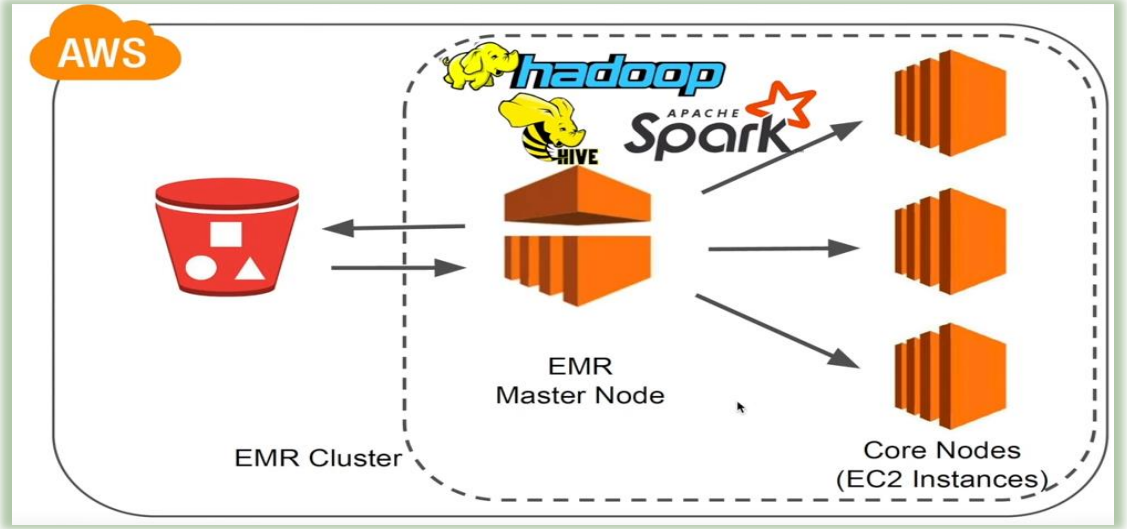


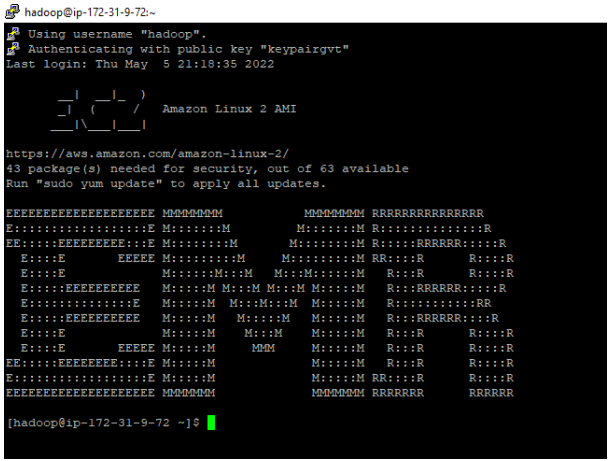


Outils utilisés pour l’analyse

Nom	Utilisation	Fonctions spécifiques
PyCharm 2021.1	Test et développement	IDE Community Edition, Debug, Synchro Git
Python 3.9.5	Moteur Python Gestionnaire de librairies	Moteur d’exécution
Pyspark **Pandas 1.4.0**	Librairie de manipulation de données Représentation des données	Manipulation de Dataframe : création, copie, filtres, tris, description, concaténation, pivotage, autre
Matplotlib 3.5.1, Seaborn 0.11.2 **Numpy 1.22.0** **Sklearn** **scipy** **Missingno**	Génération de graphiques Gestion des densités de probabilité Machine Learning	Barplot, Scatterplot, lineplot, distplot, heatmap Calcul statistique Flask, Dash : Pour la mise en ouevre de Dashboard et du simulateur de Lead

Aws Architecture

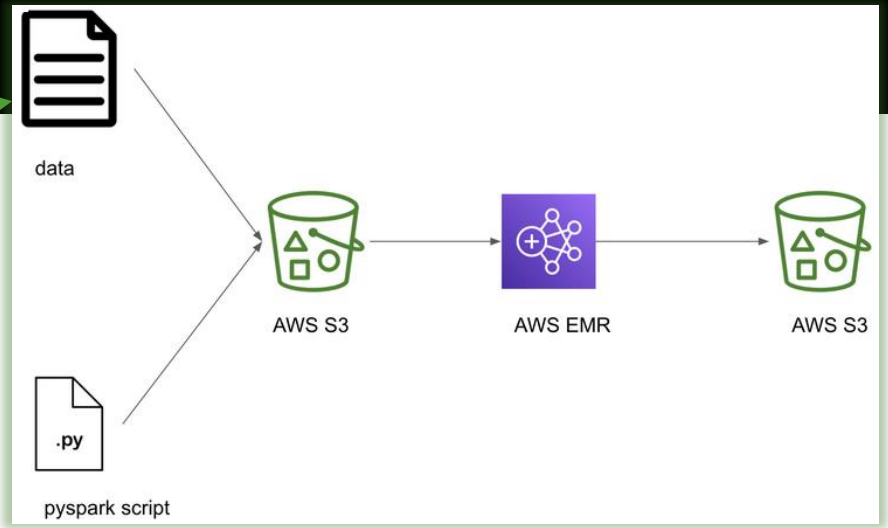





```
# Init Spark Session

spark = SparkSession \
    .builder \
    .appName("Credit_Score") \
    .getOrCreate()

# Read csv file
# By setting inferSchema=true , Spark will automatically go through the csv file and infer the
# schema of each column. This requires an extra pass over the file which will result in reading a file with inferSchema
# set to true being slower. But in return the dataframe will most likely have a correct schema given its input

S3_DATA_SOURCE_PATH = 's3://creditscoringbucket/data/lsdata_german.csv'
S3_DATA_OUTPUT_FOLDER = 's3://creditscoringbucket/output'
df = spark.read.csv(S3_DATA_SOURCE_PATH , inferSchema=True, header=True)

#df = spark.read.csv("data/data_german.csv", inferSchema=True, header=True)

# change value of score to get binary : 0 or 1
df = df.withColumn('Score', F.when(F.col("Score") == '2', 1).otherwise(0))
print(df.head())

pandasDF = df.toPandas()
print(pandasDF)
```

Execution from the master Node


```
hadoop@ip-172-31-9-72:~/creditscoring
Requirement already satisfied: python-dateutil>=2.7 in /home/hadoop/.local/lib/python3.7/site-packages (from matplotlib->missingno) (2.8.2)
Requirement already satisfied: pandas>=0.23 in /home/hadoop/.local/lib/python3.7/site-packages (from seaborn->missingno) (1.3.5)
Requirement already satisfied: typing-extensions; python_version < "3.8" in /home/hadoop/.local/lib/python3.7/site-packages (from kiwisolver>=1.0.1->matplotlib->missingno) (4.2.0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/site-packages (from python-dateutil>=2.7->matplotlib->missingno) (1.13.0)
Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.7/site-packages (from pandas>=0.23->seaborn->missingno) (2021.3)
Installing collected packages: seaborn, missingno
Successfully installed missingno-0.5.1 seaborn-0.11.2
[hadoop@ip-172-31-9-72 creditscoring]$ spark-submit ProjetCD8.py
Traceback (most recent call last):
  File "/home/hadoop/creditscoring/ProjetCD8.py", line 56, in <module>
    import findspark
ModuleNotFoundError: No module named 'findspark'
22/05/05 21:37:40 INFO ShutdownHookManager: Shutdown hook called
22/05/05 21:37:40 INFO ShutdownHookManager: Deleting directory /mnt/tmp/spark-49d745e5-4dde-48a9-ace5-d35fe23d6aae
[hadoop@ip-172-31-9-72 creditscoring]$ pip install findspark
Defaulting to user installation because normal site-packages is not writeable
Collecting findspark
  Downloading findspark-2.0.1-py2.py3-none-any.whl (4.4 kB)
Installing collected packages: findspark
Successfully installed findspark-2.0.1
[hadoop@ip-172-31-9-72 creditscoring]$ spark-submit ProjetCD8.py
```



Aws Architecture

- Après exécution du job spark « ProjetCD8.py: les images ont bien été générées prouvant la bonne exécution du traitement

image - hadoop@ec2-35-180-62-74.eu-west-3.compute.amazonaws.com - WinSCP

Local Marquer Fichiers Commandes Session Options Distant Aide

Synchroniser | File d'attente | Réglages de transfert Défaut

hadoop@ec2-35-180-62-74.eu-west-3.compute.amazonaws.com | Nouvelle session

C: Disque local | Envoyer | Éditer | Propriétés | Nouveau

C:\Users\Utilisateur\PycharmProjects\pythonProject\venv\Scripts\Data Science VAE\Credit Scoring\

Nom	Taille	Type	Date de modification
..		Répertoire parent	05/05/2022 23:26:52
image		Dossier de fichiers	05/05/2022 21:26:20
__pycache__		Dossier de fichiers	05/05/2022 20:50:38
model		Dossier de fichiers	05/05/2022 00:04:03
data		Dossier de fichiers	04/05/2022 21:56:00
log		Dossier de fichiers	27/04/2022 20:05:50
Projet CD8.py	11 KB	Python File	05/05/2022 23:29:28
helper.py	17 KB	Python File	05/05/2022 20:50:27
Logistic_KS_Charts.xlsx	5 KB	Feuille de calcul ...	05/05/2022 00:51:24
Logistic Model test.xlsx	6 KB	Feuille de calcul ...	05/05/2022 00:51:23
Logistic Model valid.x...	7 KB	Feuille de calcul ...	05/05/2022 00:51:13
Logistic Model train.x...	7 KB	Feuille de calcul ...	05/05/2022 00:51:05
metrics_calculator.py	5 KB	Python File	04/05/2022 09:58:51
zipper_function.py	2 KB	Python File	18/05/2021 20:11:16
validation_and_plots.py	5 KB	Python File	18/05/2021 20:11:16
scorecode_creator.py	6 KB	Python File	18/05/2021 20:11:16
model_builder.py	2 KB	Python File	18/05/2021 20:11:16
feature_selection.py	3 KB	Python File	18/05/2021 20:11:16
data_manipulations.py	5 KB	Python File	18/05/2021 20:11:16
build_and_execute_pi...	20 KB	Python File	18/05/2021 20:11:16

image | Rechercher des fichiers | Télécharger | Éditer | Propriétés | Nouveau

/home/hadoop/creditscoring/image/

Nom	Taille	Date de modification	Droits	Propriété...
..		05/05/2022 22:38:23	rwsrwsrwt	hadoop
Chart Null Values Plot_data.png	95 KB	05/05/2022 22:38:45	rw-rw-r--	hadoop
Features selected for modeling.png	45 KB	05/05/2022 22:38:58	rw-rw-r--	hadoop
make_confusion_matrix_chart data.png	22 KB	05/05/2022 22:39:04	rw-rw-r--	hadoop
plot_roc_Test ROC_.png	27 KB	05/05/2022 22:39:05	rw-rw-r--	hadoop
plot_roc_Train Precision-Recall curve_.png	25 KB	05/05/2022 22:39:06	rw-rw-r--	hadoop
plot_roc_Train ROC_.png	28 KB	05/05/2022 22:39:05	rw-rw-r--	hadoop
RandomForestClassifierFeatures selected for modeling.png	47 KB	05/05/2022 22:39:01	rw-rw-r--	hadoop

Stratégie d'analyse

1

Découvrir les Données

- Découverte des données Credit Scoring
- Ajout d'entête de colonne
- Taux de Remplissage par colonne
- Type de variables
- Statistiques des variables clés

2

Analyse Exploratoire / Feature Engineering

- Préparation des données
- Travaux de Nettoyage et de feature engineering
- Traitements des valeurs manquantes et imputations
- Transformation des variables catégorielles en variables numériques
- Standardisation des données
- Assemble vector

3

Modéliser / Comparer / Tester

- Modélisation -
- Résultats de performance

4

Visualisation / Interprétation / Validation

- Interprétation des Résultats

Découverte / Préparation / Analyse / Transformation	Modélisation	Validation
Helper.py ProjetCD8.py		

II. Analyse Exploratoire

Credit Scoring – Data Exploration – Cardinality Check


```python
def cardinality_calculation(df, cut_off=1):
    cardinality = df.select(*[approxCountDistinct(c).alias(c) for c in df.columns])

    ## convert to pandas for efficient calculations
    final_cardinality_df = cardinality.toPandas().transpose()
    final_cardinality_df.reset_index(inplace=True)
    final_cardinality_df.rename(columns={0: 'Cardinality'}, inplace=True)

    # select variables with cardinality of 1
    vars_selected = final_cardinality_df['index'][final_cardinality_df['Cardinality'] <= cut_off]

    return final_cardinality_df, vars_selected
```

	index	Cardinality
0	Status	4
1	Duration	33
2	History	5
3	Purpose	9
4	Amount	913
5	Savings	5
6	Employment	5
7	Rate	4
8	Situation	4
9	Debtors	3
10	Residence Since	4
11	Property	4
12	Age	53
13	Installment	3
14	Housing	3
15	Number Credits	4
16	Job	4
17	People liable	2
18	Phone	2
19	Foreign Worker	2
20	Score	2

- La cardinalité indique par colonne le nombre d’occurrence différentes par colonne dans le cas de variables numériques ou catégorielles

Variable Type

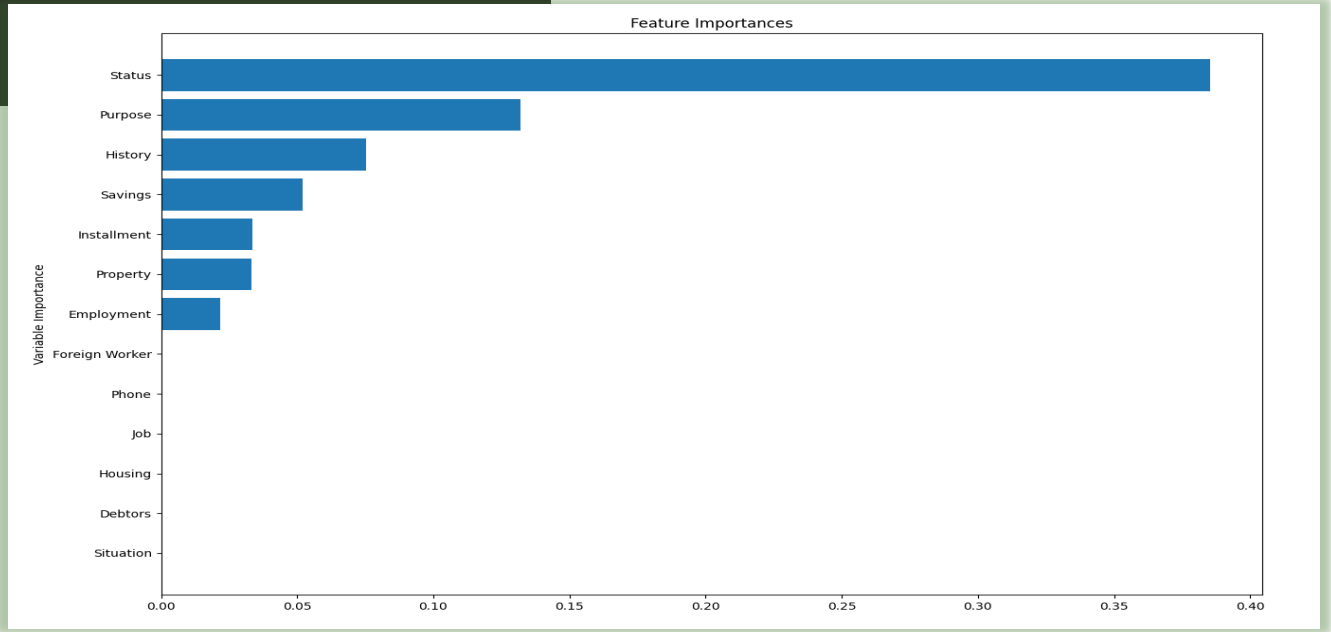

```
# 2. Identify variable type :
# Identify the data type for each variables
char_vars, num_vars = identify_variable_type(df)
print(char_vars)
print(num_vars)

# 3. cardinality check

cardinality_df, cardinality_vars = cardinality_calculation(df)
print(cardinality_df)
print(cardinality_vars)

#    4. Convert Categorical to Numerical using Label encoders
df, char_labels = category_to_index(df, char_vars)
print(char_labels)
print(df)
df.dtypes
```

```
['Status', 'History', 'Purpose', 'Savings', 'Employment', 'Situation', 'Debtors', 'Property', 'Installment', 'Housing', 'Job', 'Phone', 'Foreign Worker']
['Duration', 'Amount', 'Rate', 'Residence Since', 'Age', 'Number Credits', 'People liable', 'Score']
```

- L'identification des variables par types permet de grouper les variables par type afin de travailler sur des sous groupes de variables.

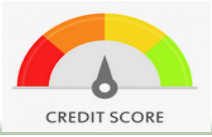
Credit – Scoring : Feature Importance – DTC Model


```
dt = DecisionTreeClassifier(featuresCol='features', labelCol=target_variable_name)
dt_model = dt.fit(df)
dt_model.featureImportances

#temporary output rf_output
dt_output = dt_model.featureImportances
features_df['Decision_Tree'] = features_df['idx'].apply(lambda x: dt_output[x] if x in dt_output.indices else 0)
print(features_df)
writeHtml(features_df, 'data/', 'features_df.html')
# Draw Importance Df features
draw_feature_importance(features_df, 'image/')
```

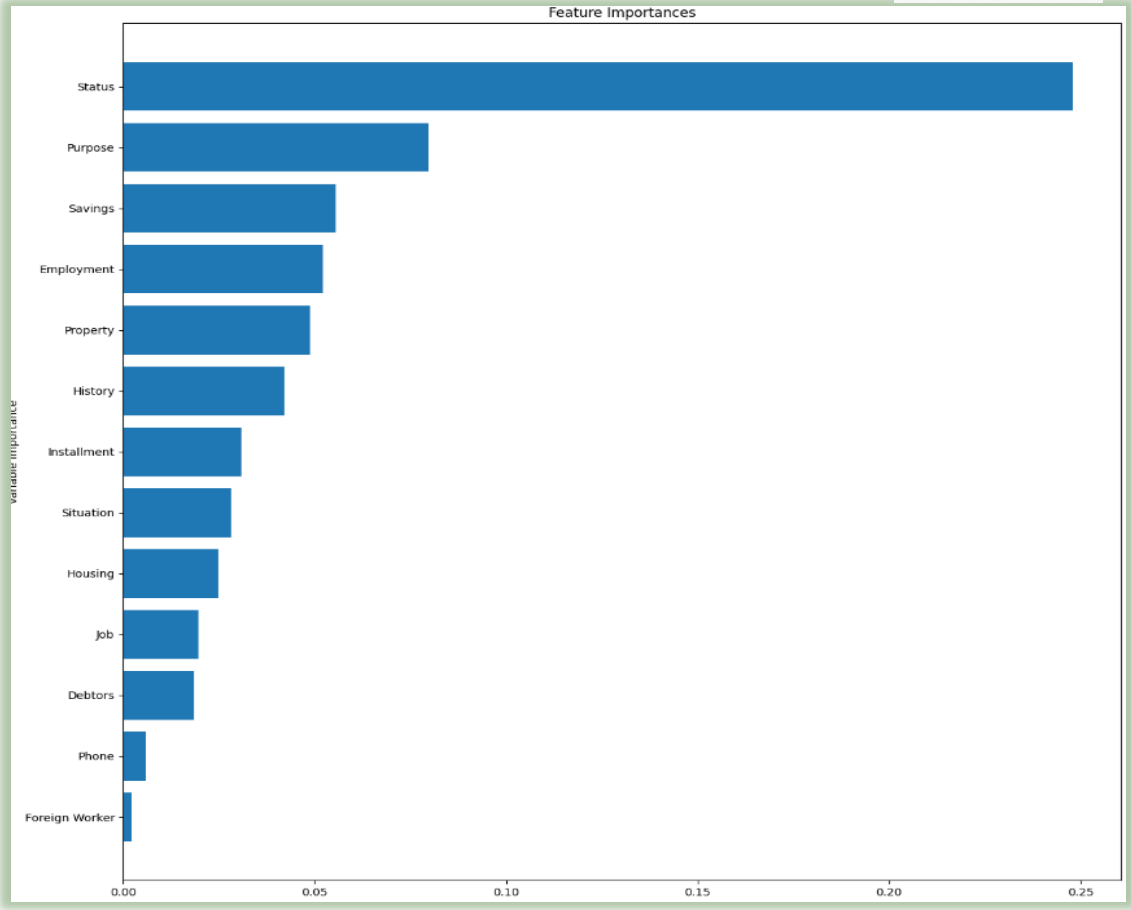


- On constate que les variables les plus importantes pour le modèle de type Arbre de Décision sont : Status, Purpose, History, Savings, Installment, Property, Employment. La méthode featureimportances permet d'obtenir ces variables.

Credit – Scoring : Feature Importance – RF Model



- On constate que les variables les plus importantes pour le modèle de type Random Forest sont : Status, Purpose, History, Savings, Installment, Property, Employment, Situation, Housing, Job, Debtors, Phone, Foreign Worker.
- La méthode featureimportances de la librarairie Scikit permet d'obtenir ces variables.
- La méthode RandomForest permet de retourner plus de variables importantes que le modèle DTC (DecisionTreeClassifier)
- Je vais partir sur une modélisation utilisant le modèle RF et voir les premiers résultats des métriques de performances



III. Modélisation

Credit Scoring – Vector Assembler


```
target_variable_name = "Score"
features_list = df.columns
dffull = df
features_list.remove(target_variable_name)
# apply the function on our dataframe

df = assemble_vectors(df, features_list, target_variable_name)
df.show()
df.schema["features"].metadata["ml_attr"]["attrs"]

for k, v in df.schema["features"].metadata["ml_attr"]["attrs"].items():
    features_df = pd.DataFrame(v)

print(features_df)
# feature importance :
```

```
+-----+--------------------+
|Score|            features|
+-----+--------------------+
|    0|[6.0,1169.0,4.0,4...|
|    1|(20,[0,1,2,3,4,5,...|
|    0|(20,[0,1,2,3,4,5,...|
|    0|[42.0,7882.0,2.0,...|
|    1|(20,[0,1,2,3,4,5,...|
|    0|[36.0,9055.0,2.0,...|
|    0|(20,[0,1,2,3,4,5,...|
|    0|(20,[0,1,2,3,4,5,...|
|    0|(20,[0,1,2,3,4,5,...|
|    1|[30.0,5234.0,4.0,...|
|    1|(20,[0,1,2,3,4,5,...|
|    1|[48.0,4308.0,3.0,...|
|    0|(20,[0,1,2,3,4,5,...|
|    1|(20,[0,1,2,3,4,5,...|
|    0|(20,[0,1,2,3,4,5,...|
|    1|(20,[0,1,2,3,4,5,...|
|    0|(20,[0,1,2,3,4,5,...|
|    0|[30.0,8072.0,2.0,...|
|    1|[24.0,12579.0,4.0...|
|    0|(20,[0,1,2,3,4,5,...|
+-----+--------------------+
only showing top 20 rows
```

- Transformateur de caractéristiques qui fusionne plusieurs colonnes en une colonne vectorielle.

Credit – Scoring : Metrics

Continuous target

- R square
- Adjusted R square
- Mean squared error
- Root mean squared error
- Mean absolute error
- Explained variance

Binary target classification

- ROC or AUC
- Accuracy
- Misclassification rate
- Precision
- Recall
- F1-score

Custom metrics

- KS Statistic
- Deciles
- Confusion matrix

Multinomial target

- Accuracy
- Misclassification rate
- Precision by label
- Recall by label
- F1-score by label
- Weighted Precision
- Weighted Recall
- Weighted F-measure

Multilabel target

- Accuracy
- Misclassification rate
- Precision
- Recall
- F1-score
- Precision by label
- Recall by label
- F1-score by label
- Hamming loss
- Subset accuracy
- Micro precision
- Micro recall
- Micro F1 measure

Recommendation/ Ranking systems

- Precision at k
- Mean average Precision
- Normalized discounted cumulative gain

❑ Dans le cas de ce projet de classification binaire : Score de Crédit : Bon payeur ou mauvais payeur potentiel, j'utiliserais les métriques : ROC, AUC

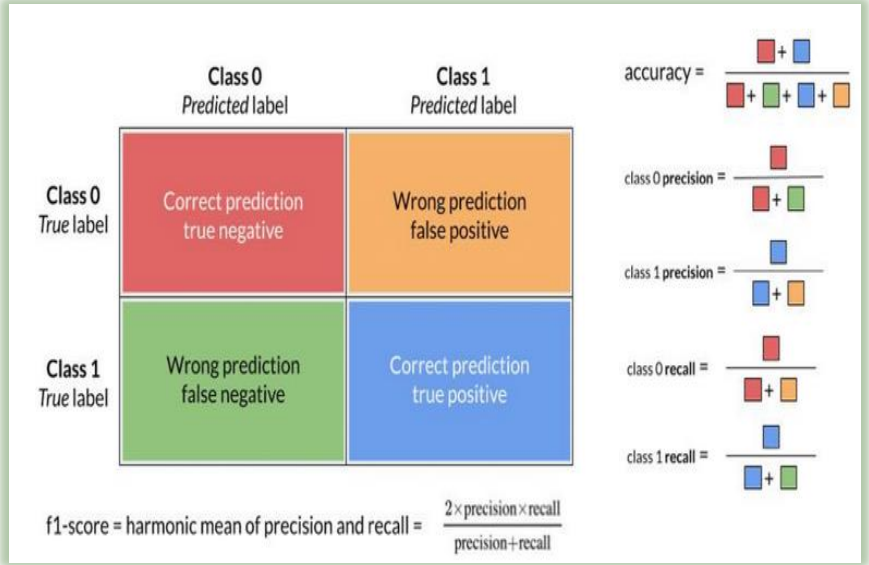
Credit Scoring – Randomforest


```
if run_randomforest_model:

    clf = RandomForestClassifier(featuresCol='features', labelCol='Score')
    clf_model = clf.fit(train)
    print(clf_model.featureImportances)
    print(clf_model.toDebugString)
    train_pred_result = clf_model.transform(train)
    test_pred_result = clf_model.transform(test)

    train_cm, train_acc, train_miss_rate, train_precision,train_recall, train_f1, train_roc, train_pr = evaluation_metrics(train_pred_result, target_variable_name)
    test_cm, test_acc, test_miss_rate, test_precision, test_recall, test_f1, test_roc, test_pr = evaluation_metrics(test_pred_result, target_variable_name)

    print('Train accuracy - ', train_acc, ', Test accuracy - ', test_acc)
    print('Train misclassification rate - ', train_miss_rate, ', Test misclassification rate - ', test_miss_rate)
    print('Train precision - ', train_precision, ', Test precision - ', test_precision)
    print('Train recall - ', train_recall, ', Test recall - ', test_recall)
    print('Train f1 score - ', train_f1, ', Test f1 score - ', test_f1)
    print('Train ROC - ', train_roc, ', Test ROC - ', test_roc)
    print('Train PR - ', train_pr, ', Test PR - ', test_pr)

    make_confusion_matrix_chart(train_cm, test_cm)

    plot_roc_pr(train_pred_result, target_variable_name, 'roc', train_roc, 'Train ROC')
    plot_roc_pr(test_pred_result, target_variable_name, 'roc', test_roc, 'Test ROC')
    plot_roc_pr(train_pred_result, target_variable_name, 'pr', train_pr, 'Train Precision-Recall curve')
```

- Le code permettant de modéliser le credit scoring utilise le modèle de type Random forest via la librairie Sciktit. Le code a été optimisé pour utiliser des fonctions réutilisables et incluses dans un script externe nommé Helper.py. Le résultat des performances sera vu dans les prochains slides.

Métriques / Performance du Modèle RF

- **Accuracy :** Taux d'erreur
- **Precision** : Le taux d'erreur, proportion d'individus mal classés doit être le plus bas possible
- **Recall** (Sensitivity) : **doit être > 0,5 pour être correct**
- **F1-Score :** Le F1-score évalue la capacité d'un modèle de classification à prédire efficacement les individus positifs, en faisant un compromis entre la precision et le recall.

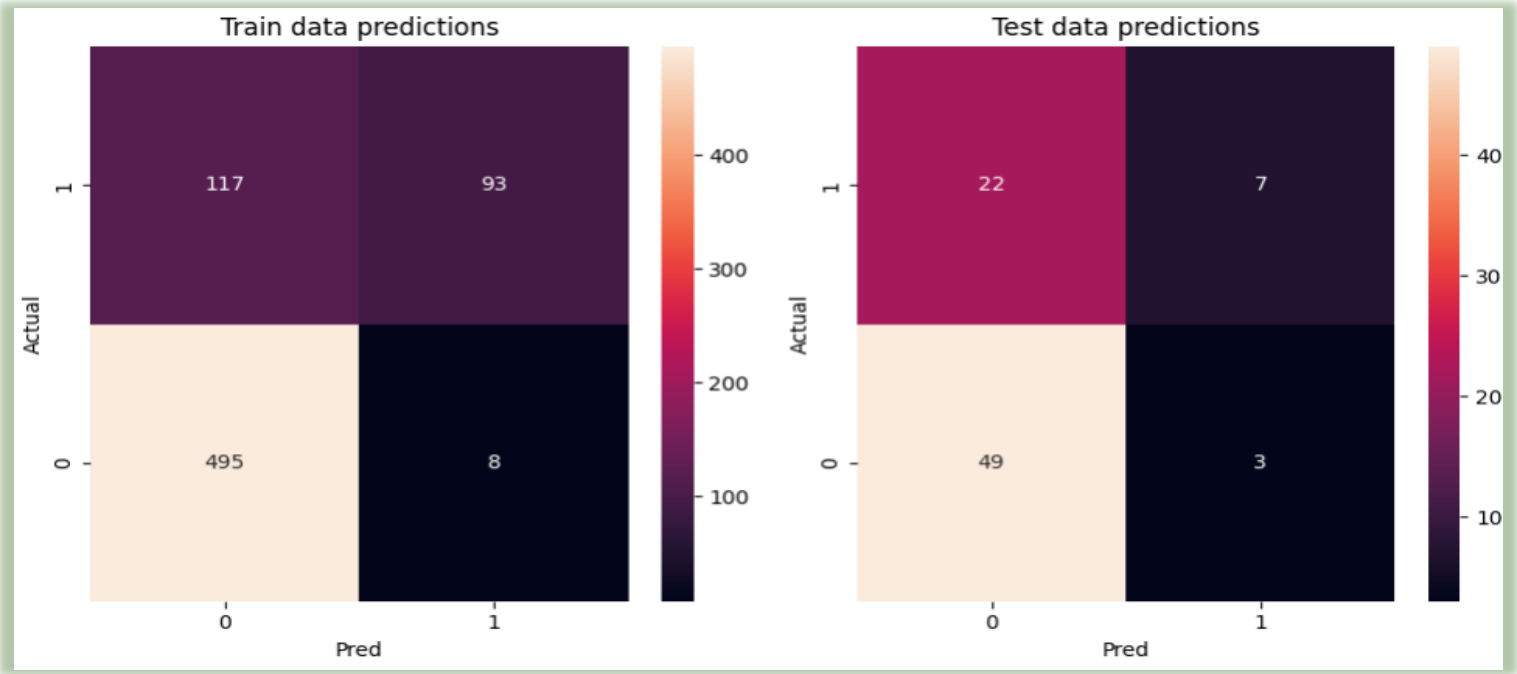



```
Train accuracy -  0.8190743338008415 , Test accuracy -  0.691358024691358
Train misclassification rate -  0.1809256661991585 , Test misclassification rate -  0.308641975308642
Train precision -  0.9263157894736842 , Test precision -  0.8333333333333334
Train recall -  0.41904761904761906 , Test recall -  0.1724137931034483
Train f1 score -  0.5770491803278688 , Test f1 score -  0.28571428571428575
Train ROC -  0.9114172110195955 , Test ROC -  0.7824933687002652
Train PR -  0.842782006425587 , Test PR -  0.6831132673618959
```

precision	Precision is the fraction of true positive examples among the examples that the model classified as positive. In other words, the number of true positives divided by the number of false positives plus true positives.
recall	Recall, also known as sensitivity, is the fraction of examples classified as positive, among the total number of positive examples. In other words, the number of true positives divided by the number of true positives plus false negatives.

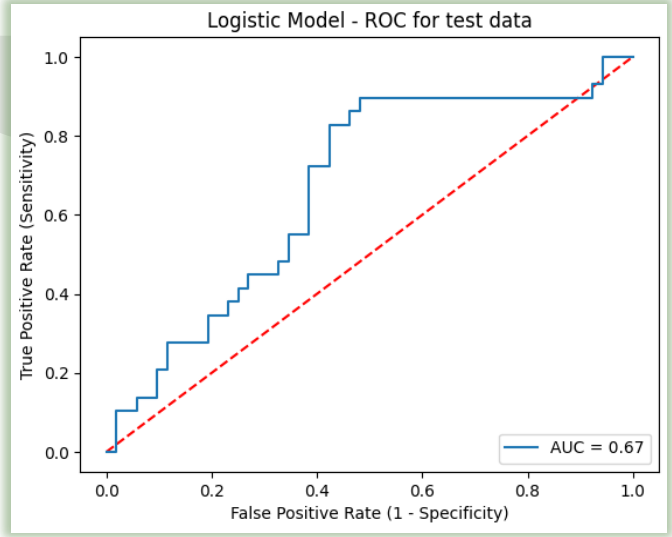
- Les résultats sont : Une précision en test de plus de 80 %

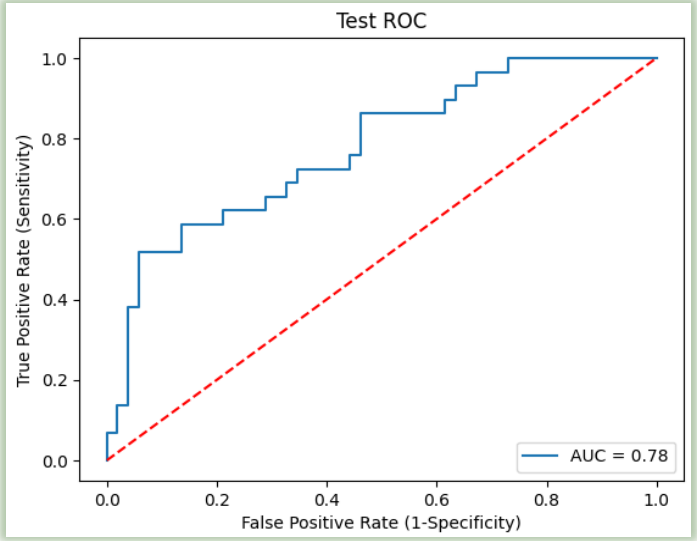
Credit Scoring – Confusion Matrix - RF



- La matrice de confusion permet de visualiser les performances des modèles d'apprentissage automatique de classification. Elle donne une meilleure idée des performances du modèle : ici le modèle de type Forêt Aléatoire (Random forest)

Credit Scoring – LR vs RF



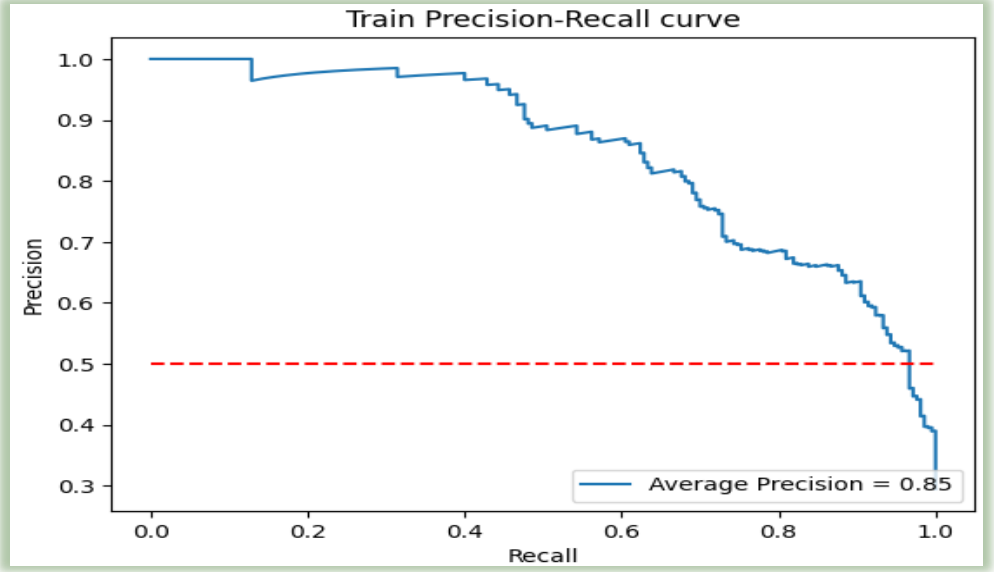



```
Train accuracy -  0.8246844319775596 , Test accuracy -  0.691358024691358
Train misclassification rate -  0.17531556802244042 , Test misclassification rate -  0.308641975308642
Train precision -  0.9207920792079208 , Test precision -  0.7
Train recall -  0.44285714285714284 , Test recall -  0.2413793103448276
Train f1 score -  0.5980707395498391 , Test f1 score -  0.358974358974359
Train ROC -  0.9116254851841324 , Test ROC -  0.7798408488063656
Train PR -  0.8389976471822906 , Test PR -  0.6674577079772207
```

- Courbe ROC (Receiver Operating Characteristic) représente l'évolution du taux du vrai positif (TPR) en fonction du taux du faux positif (FPR) en faisant varier un seuillage sur la confiance (probabilité) qu'un exemple soit dans la classe positive. Pour évaluer globalement la performance d'un modèle, on calcule l'aire sous la courbe Precision-Recall, nommée **AUC Precision-Recall**.
- **Le modèle RF est plus performant que le modèle LR car l'AUC est supérieur : 0,78 > 0,67**

Credit Scoring - RF

- Train Precision-Recall Curve




```
Train accuracy -  0.8131386861313868 , Test accuracy -  0.7155172413793104
Train misclassification rate -  0.18686131386861315 , Test misclassification rate -  0.2844827586206896
Train precision -  0.9861111111111112 , Test precision -  0.6363636363636364
Train recall -  0.35858585858585856 , Test recall -  0.19444444444444445
Train f1 score -  0.5259259259259259 , Test f1 score -  0.29787234042553196
Train ROC -  0.8975172671271237 , Test ROC -  0.734375
Train PR -  0.8304395523430775 , Test PR -  0.513561501028453
[hadoop@ip-172-31-9-72 creditscoring]$
```

V. Résultats et recommandations

Interprétation « Credit Scoring»

- J’ai pu identifier les variables les plus importantes du modèle de Credit Scoring via la librairie Scikit learn.

- Cela démontre par exemple, le rôle principal du Statut du Compte : Créditeur ou Débiteur sur l’octroi du Crédit (visuellement et sans test statistique).

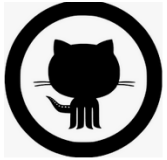
- Le modèle Random Forest nous donne des résultats meilleurs que ceux du modèle de régression Logistique : AUC 0,78 > 0,67

VI. Conclusion et Recommandations

Conclusion

- Le modèle Random Forest permet d'obtenir un taux d' AUC autour de 79% ce qui est assez correct.
- Il est possible de l'améliorer encore en cherchant à optimiser les hyperparamètres et à vérifier les performances d'autres modèles de machine Learning.
- L'intérêt ici était de vérifier le passage à l'échelle et l'exécution du traitement sur une plateforme Big data qui s'est parfaitement exécuté.
- La portabilité de cette démarche et de cette architecture à d'autres situations similaires est faisable sur un jeu de données de toute taille grâce à la scalabilité de la plateforme.

Git



Merci de votre attention