RETO ELABORADO POR LEONARDO PATIÑO RODRIGUEZ

CC: 1017186651

CORREO: [leoesleo1111@gmail.com](mailto:leoesleo1111@gmail.com)

1. **ESPECIFICACIONES**

Lenguaje: Python 3.6

anaconda-client==1.6.14

anaconda-navigator==1.8.7

anaconda-project==0.8.2

Base de datos: Mysql (XAMPP v3.2.4)

Librerias de Python

1. argparse
2. dateutil.relativedelta
3. math
4. json
5. pandas as pd
6. numpy as np
7. sklearn.model\_selection train\_test\_split
8. sklearn.neural\_network MLPClassifier
9. sklearn.metrics r2\_score
10. sklearn.metrics accuracy\_score
11. sklearn.metrics confusion\_matrix
12. sklearn.preprocessing Normalizer
13. sklearn.linear\_model LogisticRegression
14. sklearn.neighbors KNeighborsClassifier
15. sklearn.ensemble RandomForestClassifier
16. sklearn.tree DecisionTreeClassifier
17. sklearn.externals joblib
18. **PASOS PARA CORRER LOS PROGRAMAS**

1. crear entorno virtual: conda create -n hackaton python=3.6

2. Activar el entorno: conda hackaton preventivo

3. Instalar dependencias: pip install

4. Instalar mariadb version XAMPP v3.2.4

5. Iniciar servicio mysql

7. crear base de datos: CREATE DATABASE datahackaton (misma base de datos para los tres programas)

8. En la carpeta conexion/mysql.json parametrizar el string de conexión (en caso de que lo vea necesario)

9. Ejemplo de ejecución cmd

Requirements

argparse

dateutil.relativedelta

math

json

pandas as pd

numpy as np

sklearn.model\_selection train\_test\_split

sklearn.neural\_network MLPClassifier

sklearn.metrics r2\_score

sklearn.metrics accuracy\_score

sklearn.metrics confusion\_matrix

sklearn.preprocessing Normalizer

sklearn.linear\_model LogisticRegression

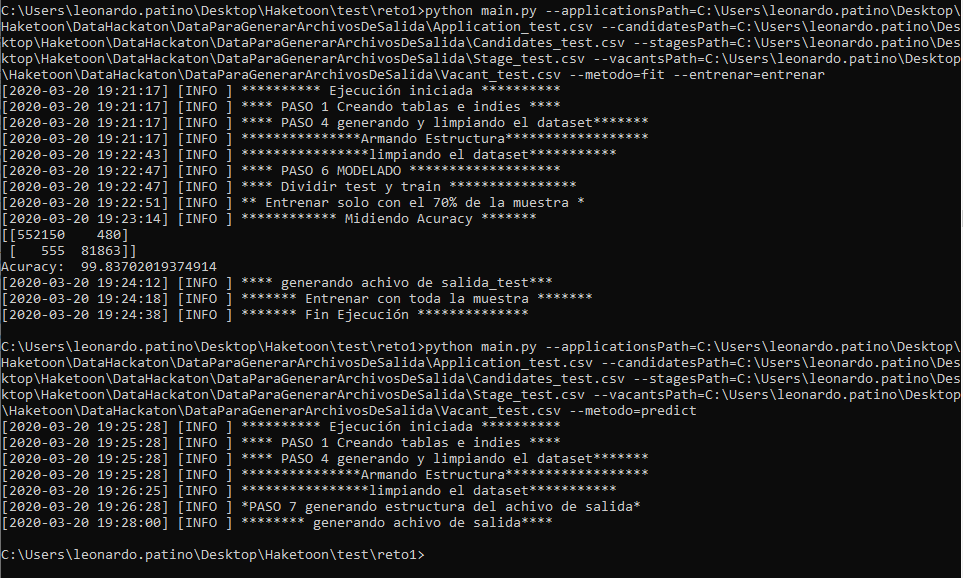
sklearn.neighbors KNeighborsClassifier

sklearn.ensemble RandomForestClassifier

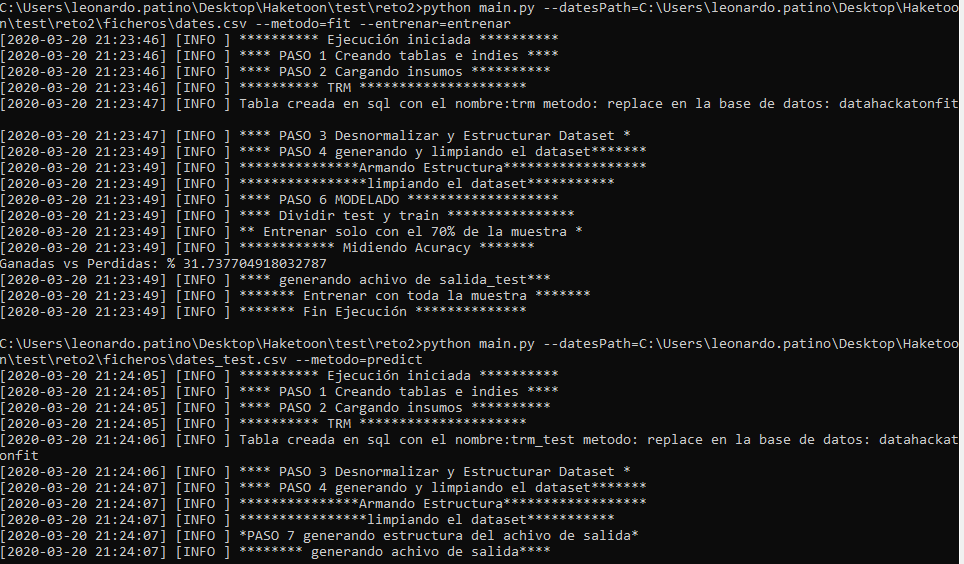
sklearn.tree DecisionTreeClassifier

sklearn.externals joblib

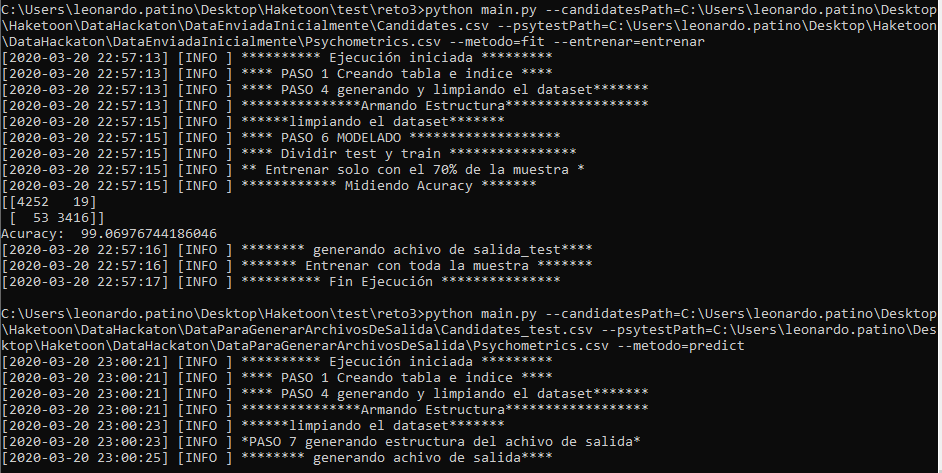
RETO 1



RETO 2



RETO 3



Como se puede observar en cada uno de los 3 programas, recibe como parámetro el método que seria : fit ó predict de acuerdo a lo solicitado, adicional recibe el parámetro entrenar que aplica solo para fit, y sirve para saber si se desea reentrenar el modelo o entrenar desde cero.

Entrenar : entrenar desde cero

Reentrenar: entrenar el modelo existente

\*\*\*\*\*RETO 1\*\*\*\*\*

#FIT

python main.py --applicationsPath=C:\Users\leonardo.patino\Desktop\Haketoon\DataHackaton\DataEnviadaInicialmente\Applications.csv --applicationstagesPath=C:\Users\leonardo.patino\Desktop\Haketoon\DataHackaton\DataEnviadaInicialmente\ApplicationStages.csv --candidatesPath=C:\Users\leonardo.patino\Desktop\Haketoon\DataHackaton\DataEnviadaInicialmente\Candidates.csv --stagesPath=C:\Users\leonardo.patino\Desktop\Haketoon\DataHackaton\DataEnviadaInicialmente\Stages.csv --vacantsPath=C:\Users\leonardo.patino\Desktop\Haketoon\DataHackaton\DataEnviadaInicialmente\Vacants.csv --metodo=fit --entrenar=entrenar

#PREDICT

python main.py --applicationsPath=C:\Users\leonardo.patino\Desktop\Haketoon\DataHackaton\DataParaGenerarArchivosDeSalida\Application\_test.csv --candidatesPath=C:\Users\leonardo.patino\Desktop\Haketoon\DataHackaton\DataParaGenerarArchivosDeSalida\Candidates\_test.csv --stagesPath=C:\Users\leonardo.patino\Desktop\Haketoon\DataHackaton\DataParaGenerarArchivosDeSalida\Stage\_test.csv --vacantsPath=C:\Users\leonardo.patino\Desktop\Haketoon\DataHackaton\DataParaGenerarArchivosDeSalida\Vacant\_test.csv --metodo=predict

\*\*\*\*\*RETO 2\*\*\*\*\*

#FIT

python main.py --datesPath=C:\Users\leonardo.patino\Desktop\Haketoon\test\reto2\ficheros\dates.csv --metodo=fit --entrenar=entrenar

#PREDICT

python main.py --datesPath=C:\Users\leonardo.patino\Desktop\Haketoon\test\reto2\ficheros\dates\_test.csv --metodo=predict

\*\*\*\*\*RETO 3\*\*\*\*\*

#FIT

python main.py --candidatesPath=C:\Users\leonardo.patino\Desktop\Haketoon\DataHackaton\DataEnviadaInicialmente\Candidates.csv --psytestPath=C:\Users\leonardo.patino\Desktop\Haketoon\DataHackaton\DataEnviadaInicialmente\Psychometrics.csv --metodo=fit --entrenar=entrenar

#PREDICT

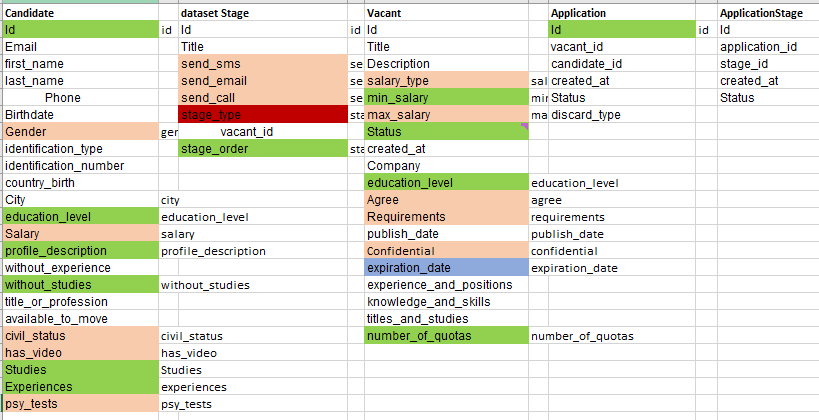
python main.py --candidatesPath=C:\Users\leonardo.patino\Desktop\Haketoon\DataHackaton\DataParaGenerarArchivosDeSalida\Candidates\_test.csv --psytestPath=C:\Users\leonardo.patino\Desktop\Haketoon\DataHackaton\DataParaGenerarArchivosDeSalida\Psychometrics.csv --metodo=predict

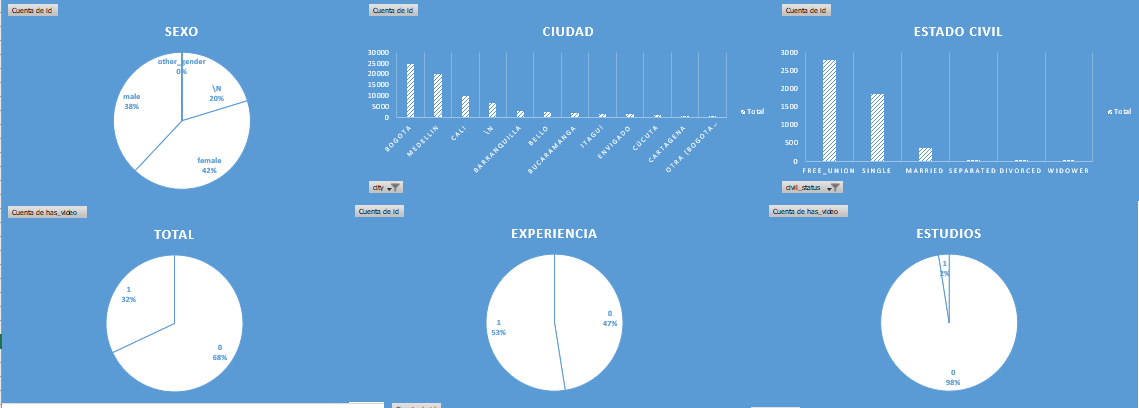
1. **ANALISIS Y DESCRIPCIÓN DE LOS MODELOS**

Se realiza varios análisis descriptivos para conocer la naturaleza de los datos:

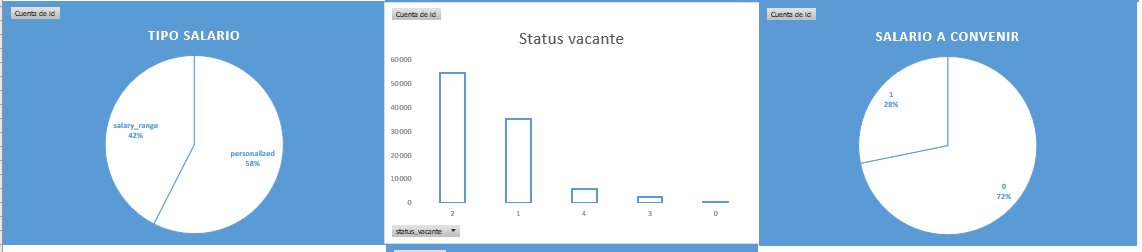
Reto 1

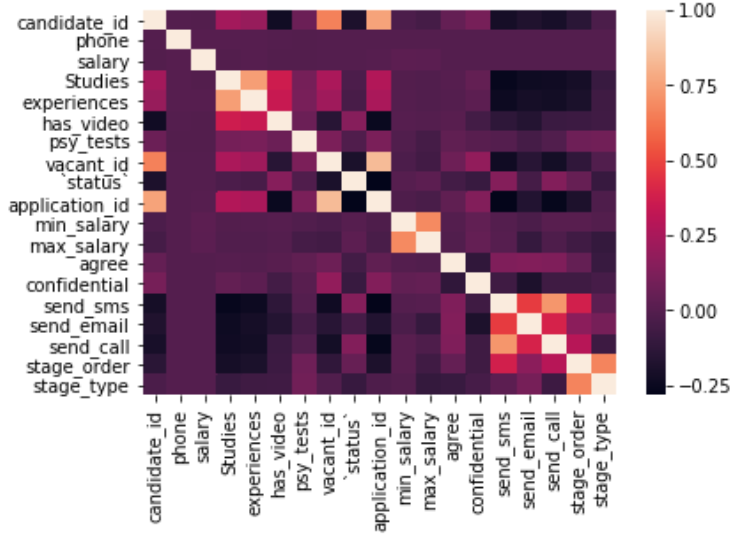
Observar posibles variables significativas



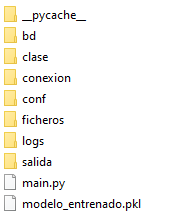








El programa tiene la siguiente arquitectura



**Bd:** contiene una clase bd.py con métodos para la base de datos

**Clase**: contiene dos clases muy utilices (análisis.py – limpieza.py)

**Conexión**: contiene archivo json parametrizable para el string de conexión

**Conf**: contiene una clase logger para el manejo de logs de errores – info

**Ficheros**: contiene archivos de entrada al modelo (opcional)

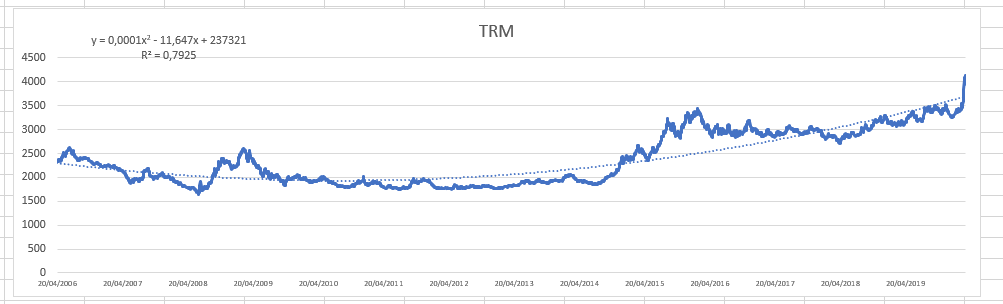
**Logs**: se almacenen los logs generados por la clase logger

**Salida**: contiene las salidas del programa

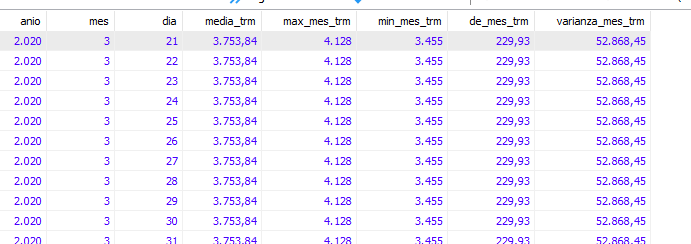
**Manin.py** : archivo principal del programa en donde se inicia

**modelo\_entrenado.pkl**: modelo que genera el programa

reto 2



Se generan nuevas variables de estacionalidad



Contiene la misma arquitectura del programa 1

Reto 3

Contiene la misma arquitectura del programa 1

Modelo implementado:

Se prueban con varios modelos y se deja el que mejor score y acuracy tiene.

