



# Python for Data Analysis Project

By Raphaël RICHARD and Thibaut  
ROUET

Our dataset :

**Diabetes stats in 130 US  
hospitals between 1999  
and 2008**



# Problem



Look into the readmission potential of a patient



Search for other parameters that could be interesting



Very large dataset, a purge of the useless data is needed



## First things to do : Filtering

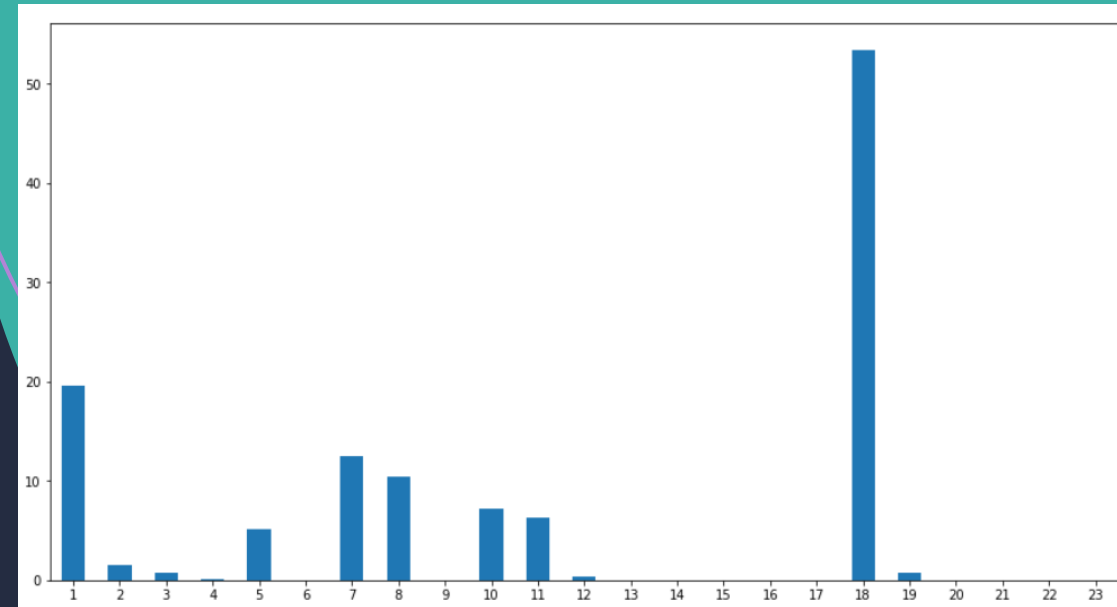
- Look for almost empty columns and remove them
- Replace all the undefined values like « ? » and « None » by NaN to remove them while still making them count
- Check all the different possible treatments and transform them into simpler columns

It gives us a new dataset easier to study,  
as we remove around 40% of useless data

	patient_nbr	race	gender	age	admission_type_id	discharge_disposition_id	admission_source_id	time_in_hospital	medical_specialty	num_lab_procedures	...	A1Cresult	change	diabetesMed	readmitted	T1	T2	T3	T4	T5	T6
0	8222157	Caucasian	Female	[0-10)	6	25	1	1	Pediatrics-Endocrinology	41	...	NaN	No	No	0	NaN	NaN	NaN	NaN	NaN	NaN
1	55629189	Caucasian	Female	[10-20)	1	1	7	3	NaN	59	...	NaN	Ch	Yes	0	18-Up	NaN	NaN	NaN	NaN	NaN
2	86047875	AfricanAmerican	Female	[20-30)	1	1	7	2	NaN	11	...	NaN	No	Yes	0	7-Steady	NaN	NaN	NaN	NaN	NaN
3	82442376	Caucasian	Male	[30-40)	1	1	7	2	NaN	44	...	NaN	Ch	Yes	0	18-Up	NaN	NaN	NaN	NaN	NaN
4	42519267	Caucasian	Male	[40-50)	1	1	7	1	NaN	51	...	NaN	Ch	Yes	0	7-Steady	18-Steady	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
101761	100162476	AfricanAmerican	Male	[70-80)	1	3	7	3	NaN	51	...	>8	Ch	Yes	0	1-Steady	18-Down	NaN	NaN	NaN	NaN
101762	74694222	AfricanAmerican	Female	[80-90)	1	4	5	5	NaN	33	...	NaN	No	Yes	0	18-Steady	NaN	NaN	NaN	NaN	NaN
101763	41088789	Caucasian	Male	[70-80)	1	1	7	1	NaN	53	...	NaN	Ch	Yes	0	1-Steady	18-Down	NaN	NaN	NaN	NaN
101764	31693671	Caucasian	Female	[80-90)	2	3	7	10	Surgery-General	45	...	NaN	Ch	Yes	0	7-Steady	10-Steady	18-Up	NaN	NaN	NaN
101765	175429310	Caucasian	Male	[70-80)	1	1	7	6	NaN	13	...	NaN	No	No	0	NaN	NaN	NaN	NaN	NaN	NaN

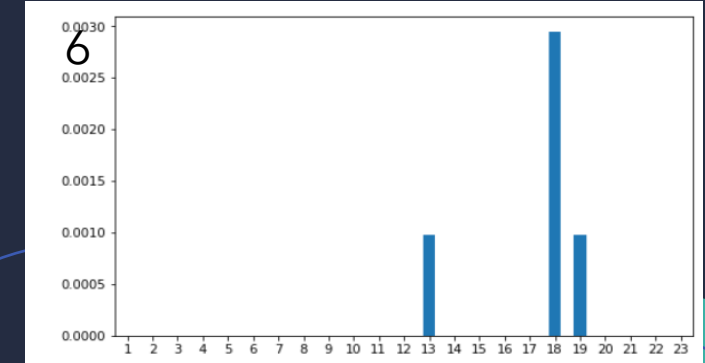
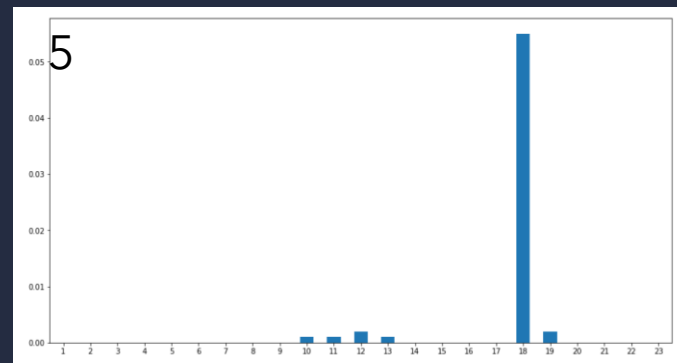
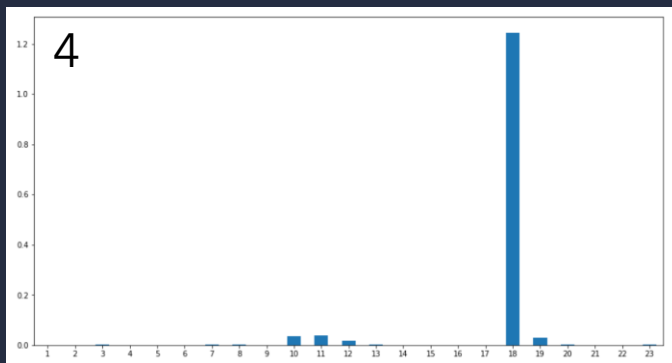
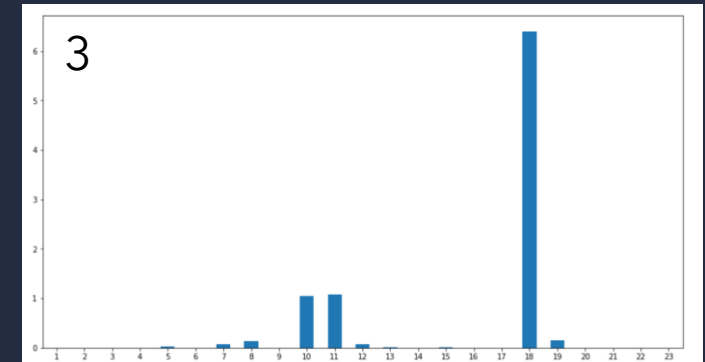
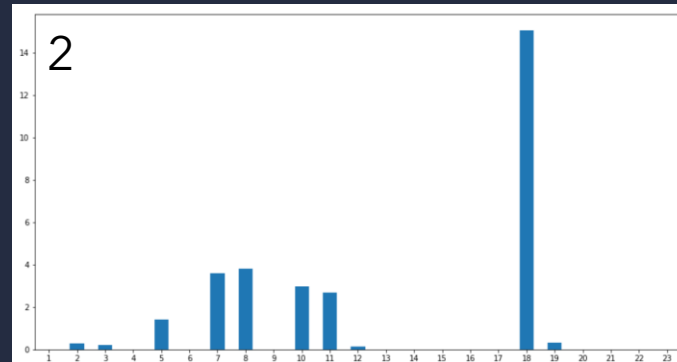
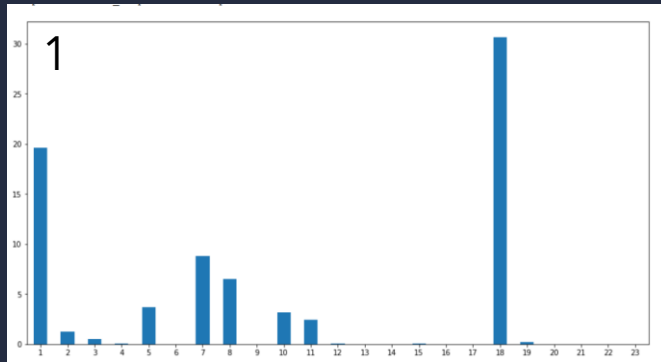
We decide to look into the percentage of deliverance of all the different treatments

- It gives us the following graphic :
- The 18th treatment is used in more than 50% of the cases
- It seems logical as it is the insulin

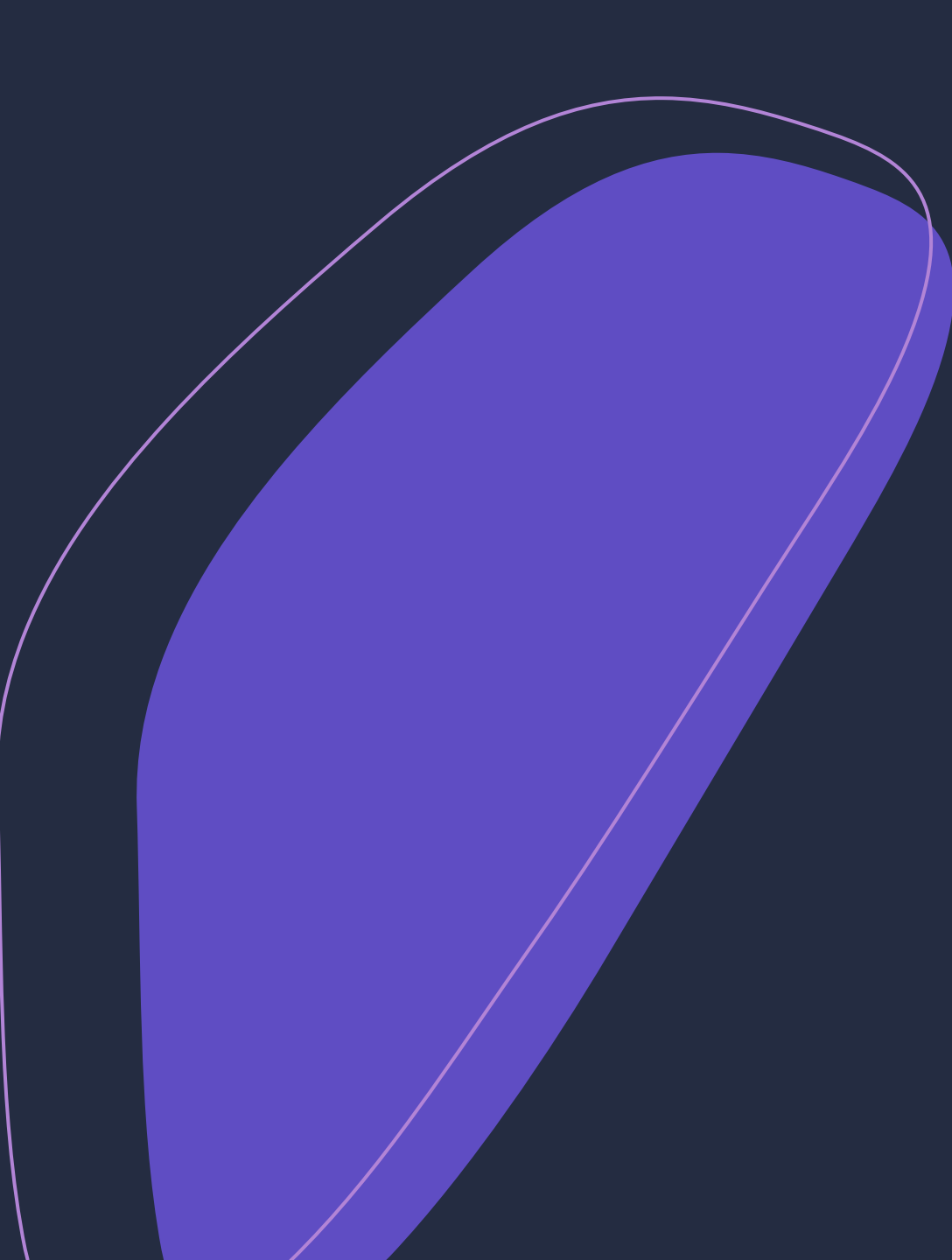


# We then looked at if it changed when we looked only at the 1st, 2<sup>nd</sup>, ..., 6th treatment

- We got those graphics:







We also decided to take a look at what treatment was used after the insulin when needed

- We find out that the treatments 19, 20, 21 and 22 are used when the insulin proved inefficient.
- Those numbers correspond to glyburide-metformin, glipizide-metformin, glimepiride-pioglitazone and metformin-rosiglitazone"

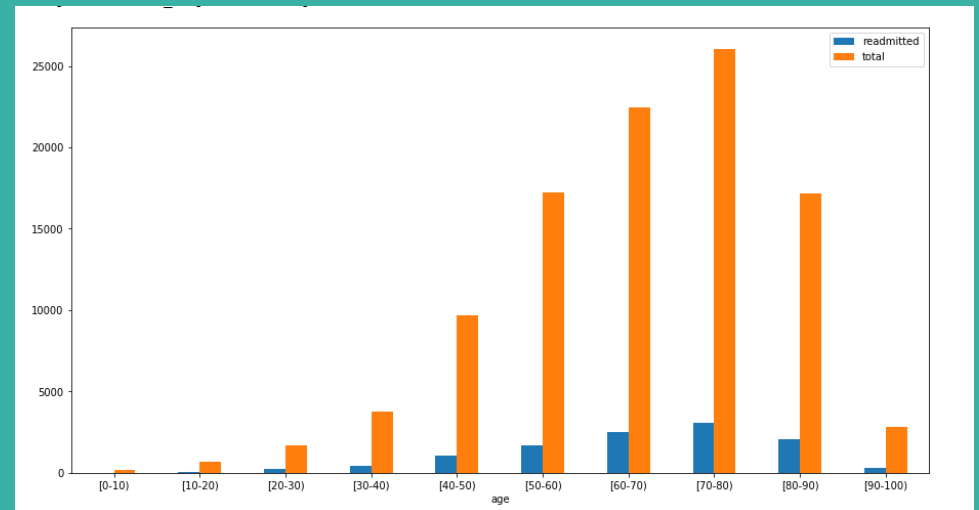


We finally decided to look into the readmission need under 30 days, by linking it to the patient's age

We found out this number of person getting readmitted compared to the number of patient by age

	readmitted	total
age		
[0-10)	3	161
[10-20)	40	691
[20-30)	236	1657
[30-40)	424	3775
[40-50)	1027	9685
[50-60)	1668	17256
[60-70)	2502	22483
[70-80)	3069	26068
[80-90)	2078	17197
[90-100)	310	2793

It allowed us to  
make the following  
graphic to represent  
it

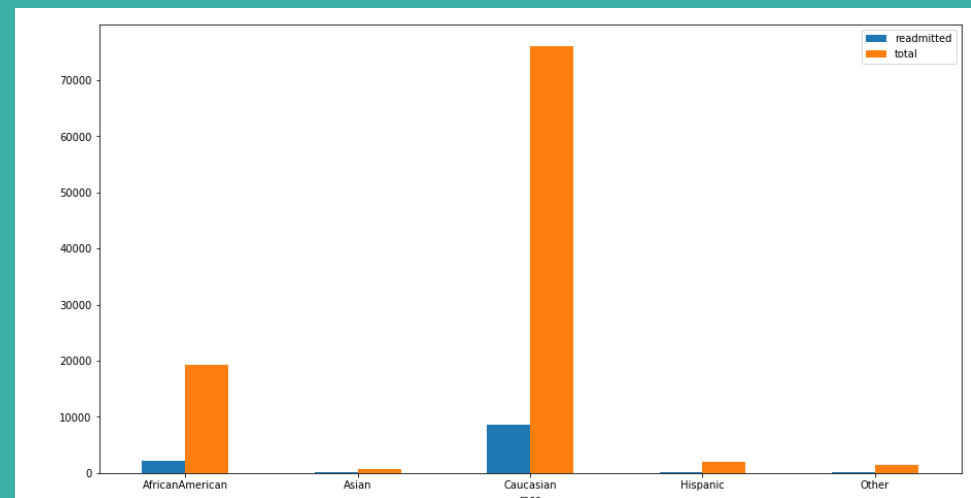


As the age seems to have an impact on the readmission, we decided to look onto the race

- We mainly did it because we know that the question of race is an important one in the US, and that it could have impacted the readmission of some people

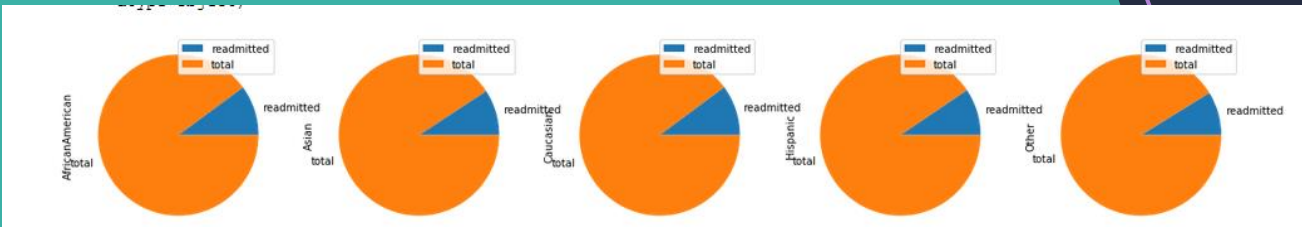


We got this barchart showing us the number of people readmitted for each ethny



However, as it wasn't representative enough, we decided to look into the proportion of readmitted people by race

- It appears that the proportion is the same no matter the ethnicity, so the race column isn't pertinent for our study and we remove it



Finally we decide to try and predict whether someone would be readmitted based on all their characteristics.

- We start by transforming the dataset so that every string value becomes a float, as the different models can't consider string values

```
def unique_map(dataframe: pd.DataFrame, col: str):
    Tn = ["T1", "T2", "T3", "T4", "T5", "T6"]
    if col in treatments:
        values = ['No', 'Steady', 'Down', 'Up']
    elif col in ['diag_1', 'diag_2', 'diag_3']:
        values = list(set(df[col].unique().astype("str").tolist() + df[col].unique().astype("str").tolist()))
        values.sort()
    elif col in Tn:
        values = []
        for T in Tn:
            values += df[T].unique().astype("str").tolist()
        values = list(set(values))
        values.sort()
    else:
        values = dataframe[col].unique().astype("str").tolist()
        values.sort()
    for val in values:
        dataframe[col] = dataframe[col].replace(val, values.index(val))

df1 = df.copy()
cols = ["gender", "age", "medical_specialty", "AlCresult", "change", "diabetesMed", 'diag_1', 'diag_2', 'diag_3'] + treatments
for col in cols:
    unique_map(df1, col)
df1 = df1.drop(columns=["race"])
df1 = df1.replace(np.nan, -1)
df1
```

```
] : from sklearn import preprocessing
    df1 = df1.replace(np.nan, "None")
    X = df1.drop(columns=["readmitted"])
    Y = df1[['readmitted']]
```

```
] : from sklearn.model_selection import train_test_split
    X_train, X_test, Y_train, Y_test = train_test_split(X,Y)
```

```
] : from sklearn.ensemble import RandomForestClassifier
    modell = RandomForestClassifier()
    modell.fit(X_train, Y_train)
    Y_pred = modell.predict(X_test)
```

```
C:\Users\thiba\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:245: FutureWarning: The default value of n_estimators
will change from 10 in version 0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
C:\Users\thiba\Anaconda3\lib\site-packages\ipykernel_launcher.py:3: DataConversionWarning: A column-vector y was passed whe
n a 1d array was expected. Please change the shape of y to (n_samples,), for example using ravel().
  This is separate from the ipykernel package so we can avoid doing imports until
```

```
] : from sklearn.metrics import accuracy_score
    accuracy_score(Y_test, Y_pred)
```

```
0.8884521657102429
```

# We really start our study by doing a RandomForest

It seems pretty precise, with an accuracy of 88,8%



## We then try a Decision Tree

- It seems less precise than the RandomForest

### Decision Tree

```
134]: from sklearn import tree
      model2 = tree.DecisionTreeClassifier()
      model2.fit(X_train, Y_train)
      Y_tree = model2.predict(X_test)

135]: accuracy_score(Y_test, Y_tree)

: 0.7986400440216964
```

```
: from sklearn.neighbors import KNeighborsClassifier
modelKNN = KNeighborsClassifier()
modelKNN.fit(X_train, Y_train)
Y_KNN = modelKNN.predict(X_test)
```

```
C:\Users\thiba\Anaconda3\lib\site-packages\ipykernel_launcher.py:1:
n a 1d array was expected. Please change the shape of y
This is separate from the ipykernel package so we can
```

```
: accuracy_score(Y_test, Y_KNN)
```

```
0.8818489112491157
```

# We keep going with a KNN classifier

It appears to be almost as  
good as the RandomForest

# We continue with a Support Vector

We have here our most precise  
model yet

## Support Vector

```
from sklearn.svm import SVC
modelSVM = SVC()
modelSVM.fit(X_train, Y_train)
Y_SVM = modelSVM.predict(X_test)
```

✓ 1m 49.5s

Python

C:\Users\raph1\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\utils\validation.py:993: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n\_samples, ), for example using ravel().  
y = column\_or\_1d(y, warn=True)

```
accuracy_score(Y_test, Y_SVM)
```

✓ 0.3s

Python

0.8884914707963211

# Finally, we try a GridSearch

- We couldn't make it to the end of the process because of the size of our dataset, it would have taken us way to much hours

```
from sklearn.model_selection import GridSearchCV
param_grid = [
    {'C': [1, 10], 'kernel': ['linear']}
]
#{'C': [1, 10, 100, 1000], 'gamma': [0.001, 0.0001], 'kernel': ['rbf']},
grid = GridSearchCV(SVC(), param_grid, scoring='accuracy', n_jobs=-1, verbose=3)
grid.fit(X_train, Y_train)
print (grid.best_score_, grid.best_estimator_)

11m 9.5s Python
```

Fitting 5 folds for each of 2 candidates, totalling 10 fits