10 JANUARY 2021
MICHEL FUVEL IBO 2

YEAR PREDICTION MILLION
MILLION SONGS DATASET

MESIGI595120 PYTHON FOR DATA ANALYSIS

ESILV
ENGINEERING SCHOOL
DE VINCI PARIS

OUR PROJECT

THE GOAL OF THIS PROJECT IS TO PREDICT THE YEAR WHERE A SONG WAS COMPOSED BY IT'S MUSICAL FEATURES

DATASET INFORMATION

- This Dataset is made out of 515 345 different records from 1920 to 2011 included
- Each record has 91 different features where the first one is the year when the song was published and the following onces are related to the various timbres of each song

ESILV
ENGINEERING SCHOOL
DE VINCI PARIS

SUMMARY

DATA PROCESSING

MACHINE LEARNING MODELS

FLASK API

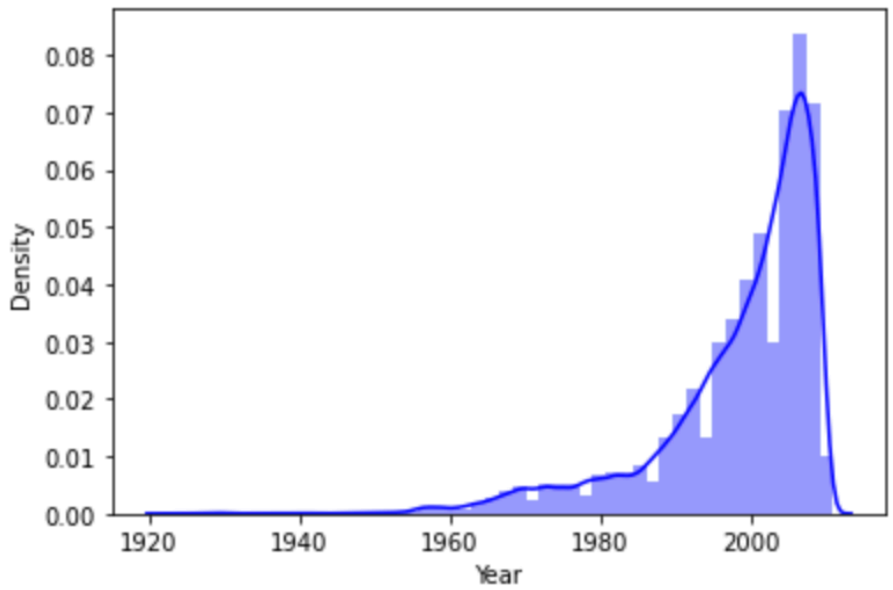
ESILV
ENGINEERING SCHOOL
DE VINCI PARIS

DATA PROCESSING

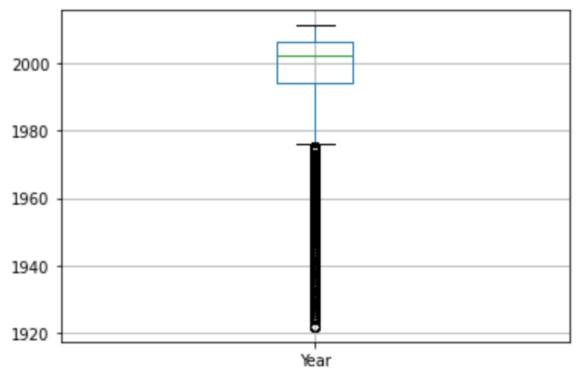
When we analyse the data, we realise that the distribution of the dataset is very unequal.

From 1920 till 1960 there are no more than an average of a 100 songs published by year while in 2007, a total of 39.404 songs where published.

We count 515 345 records for a mean around the year 1998



	Year
count	515345.000000
mean	1998.397082
std	10.931046
min	1922.000000
25%	1994.000000
50%	2002.000000
75%	2006.000000
max	2011.000000



ESILV
ENGINEERING SCHOOL
DE VINCI PARIS

MACHINE LEARNING MODELS

We tried Several different models but any of the accuracies that we were getting in return were good enough.

- Decision Tree
- Random Forest Regressor
- KNeighbors Classifier

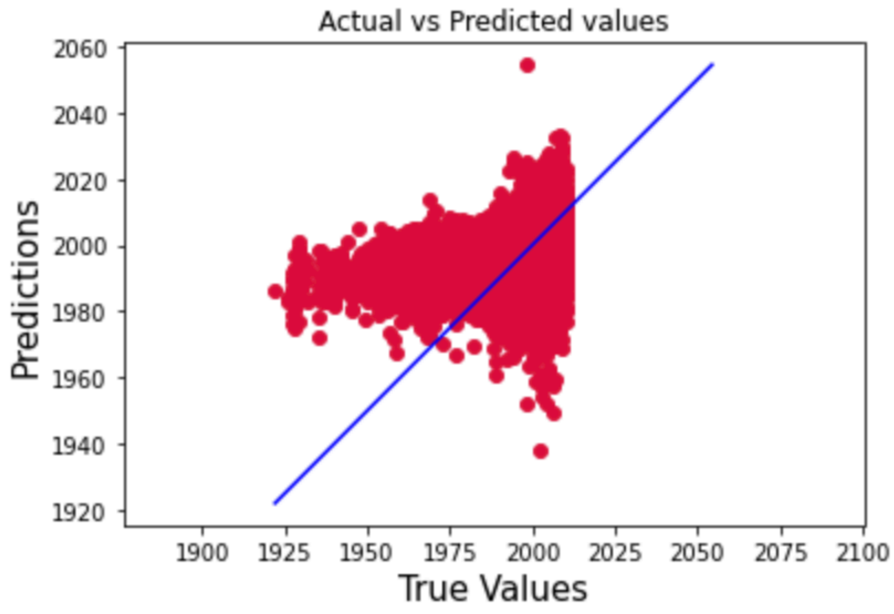
```
In [48]: #Modele Decision Tree

         Dtree = tree.DecisionTreeClassifier()
         Dtree.fit(x_train, y_train)

Out[48]: DecisionTreeClassifier()

In [49]: Dtree.score(x_test, y_test)

Out[49]: 0.0737079044135482

In [50]: Modele Random Forest Regressor

         from sklearn.ensemble import RandomForestRegressor
         rf = RandomForestRegressor().fit(x_train, y_train)
         rf.score(x_test, y_test)

Out[50]: 0.3097766124229031
```

```
In [84]: #Modele Linear Regression

         reg = LinearRegression().fit(x_train, y_train)
         reg.score(x_test, y_test)

Out[84]: 0.23988906124412013

In [85]: y_pred = reg.predict(x_test)

In [86]: # Plot outputs
         plt.scatter(y_test, y_pred, c='crimson')
         #plt.scatter(x_train, y_train, color='blue', linewidth=3)


         p1 = max(max(y_pred), max(y_test))
         p2 = min(min(y_pred), min(y_test))
         plt.plot([p1, p2], [p1, p2], 'b-')
         plt.xlabel('True Values', fontsize=15)
         plt.ylabel('Predictions', fontsize=15)
         plt.title("Actual vs Predicted values")
         plt.axis('equal')
         plt.show()
```

MACHINE LEARNING MODELS

Linear Regression

```
In [84]: #Modele Linear Regression

         reg = LinearRegression().fit(x_train, y_train)
         reg.score(x_test, y_test)

Out[84]: 0.23988906124412013

In [85]: y_pred = reg.predict(x_test)

In [86]: # Plot outputs
         plt.scatter(y_test, y_pred, c='crimson')
         #plt.scatter(x_train, y_train, color='blue', linewidth=3)



         p1 = max(max(y_pred), max(y_test))
         p2 = min(min(y_pred), min(y_test))
         plt.plot([p1, p2], [p1, p2], 'b-')
         plt.xlabel('True Values', fontsize=15)
         plt.ylabel('Predictions', fontsize=15)
         plt.title("Actual vs Predicted values")
         plt.axis('equal')
         plt.show()
```



MACHINE LEARNING MODELS

Réseau de Neurones

In [24]:
```
#Y(test and train) de 0 à 90 les annees 1922 - 2011

Y_train = np_utils.to_categorical(y_train, num_attributes)
Y_test = np_utils.to_categorical(y_test, num_attributes)
```

In [43]:
```
# Reseau de Neurones

batch_size = 5000 # un tres petit pourcentage du Dataset ( 500.000 donnees )
num_epochs = 10


model = Sequential()
model.add(Dense(input_dim=num_attributes, units=5000, activation='relu'))
model.add(Dense(units=1))

model.compile(optimizer='rmsprop', loss='mse', metrics=['accuracy'])
history = model.fit(X_train, Y_train,
                    validation_data=(X_test, Y_test),
                    batch_size=batch_size,
                    epochs=num_epochs)
```

In [47]:
```
loss, acc = model.evaluate(x_test, Y_test, verbose=0)
print(acc)
```

```
0.934000551700592
```