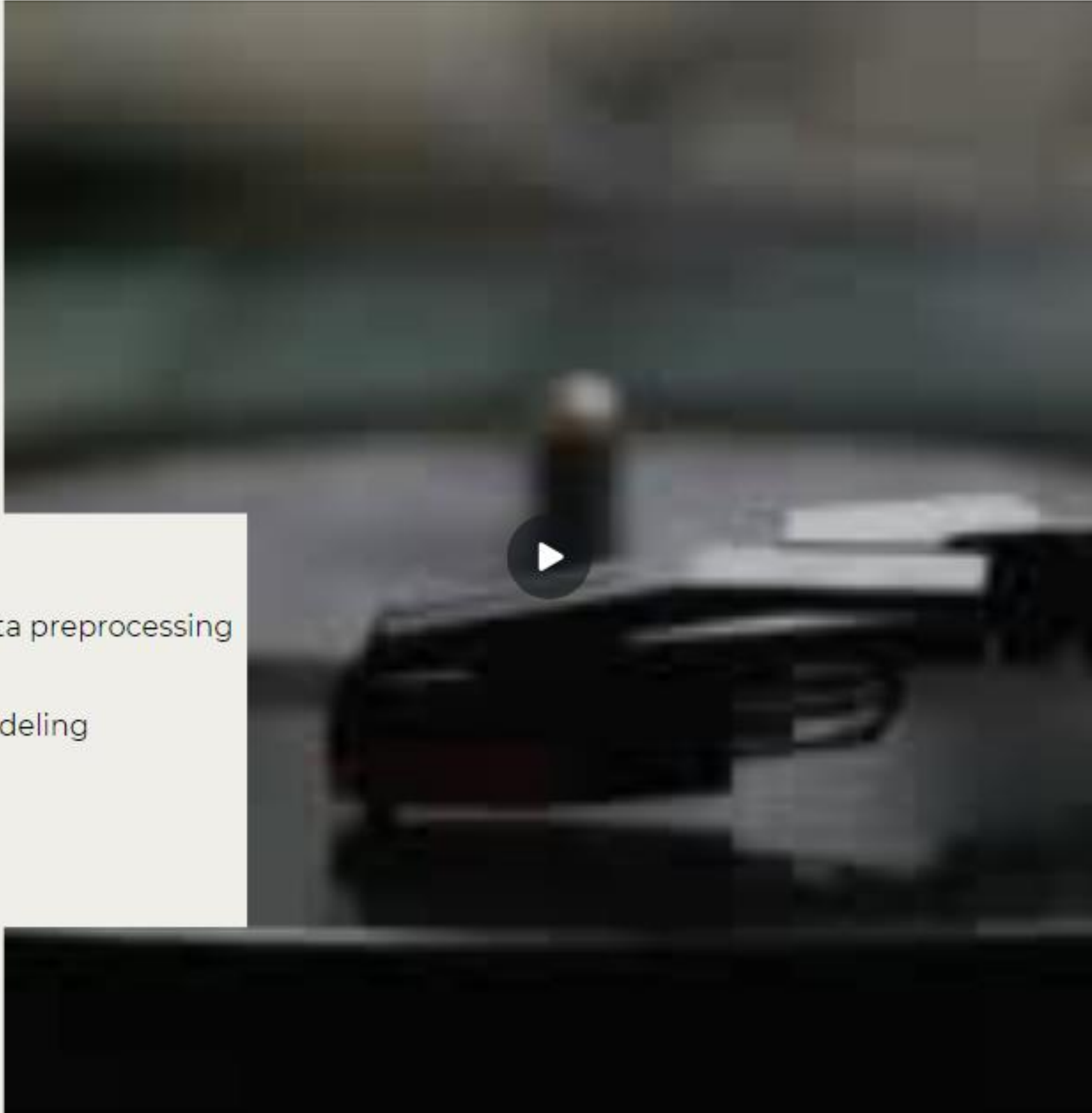
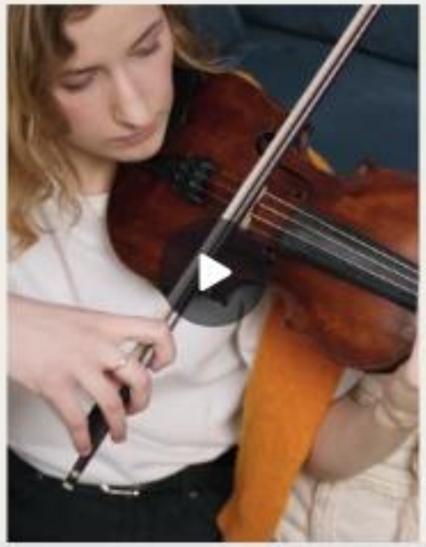
PREDICTING RELEASE YEAR OF SONGS

Presented by :
Thibault Bilville
Meriem Boussaadia
Mehdi Jegham

AGENDA



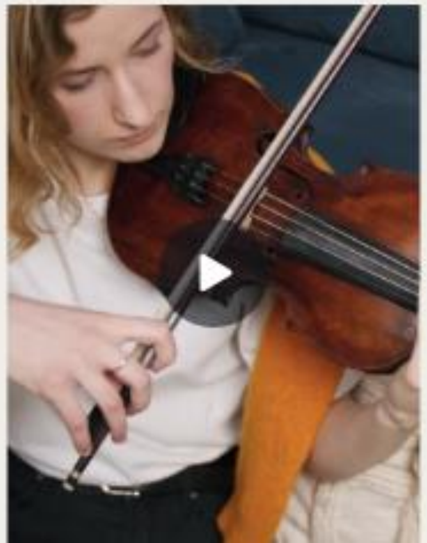


INTRODUCTION

In the fascinating realm where music and technology converge, our project takes a unique turn—predicting the release years of songs based on their audio features. Utilizing a subset of the Million Song Dataset (MSD), our objective is to uncover the temporal essence embedded in musical compositions.

The project fits into the broader context of music analysis and understanding the evolution of musical styles over time.
By utilizing the Million Song Dataset, which covers a substantial period from 1922 to 2011, the study captures a diverse range of songs reflecting changing trends in popular music.

RELEASE YEAR PREDICTION ON MSD



- The specific task of predicting release years directly ties into the overarching goal of understanding the temporal dynamics of music.
- By focusing on timbral features, the study delves into the sonic characteristics that contribute to the uniqueness of songs across different eras.

GOALS

Goals 1

Temporal Prediction Accuracy:

- Develop a robust machine learning model to accurately predict the release year of songs.
- Achieve a high level of precision in capturing the temporal nuances embedded in audio features.

Goals 2

Feature Analysis and Selection:

- Explore the relevance of different audio features in predicting song release years.
- Identify key timbral elements that contribute significantly to the predictive power of the model.

Goals 3

Cultural and Musical Insights:

- Uncover cultural and musical trends reflected in the audio features of songs from different periods.
- Gain insights into how timbral characteristics evolve over time in the music landscape.

ASKED QUESTION

The central question revolves around predicting the release year of songs, a task with inherent challenges given the diverse and evolving nature of music over the years.

Timbre, representing the unique sound qualities of each song, is chosen as a predictor, indicating an assumption that timbral characteristics might capture the temporal evolution of musical styles.

DATASET

- **Rows** : 515,345
- **Columns** : 91

year:

- This is the target variable representing the release year of the song.

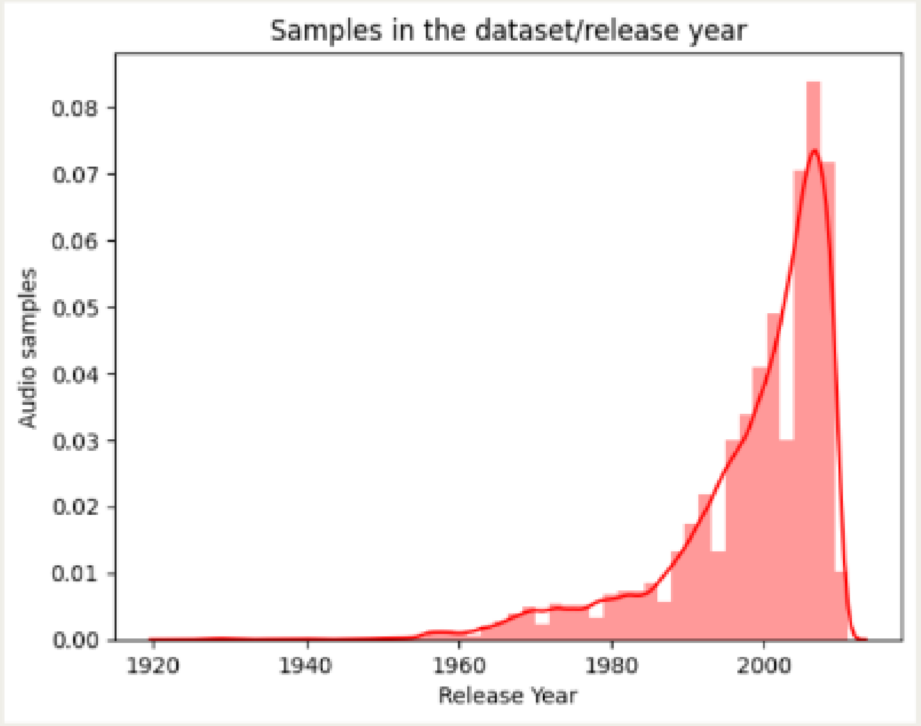
TimbreAvg1 to TimbreAvg78:

- These columns likely represent the average values of different timbre features across the audio track. Timbre generally refers to the quality of a musical note or sound that distinguishes different types of sound production.

TimbreCovariance1 to TimbreCovariance78:

- These columns may represent the covariance values of different timbre features. Covariance provides insights into how two features change together.

- **Time Range** : Songs are mostly western, commercial tracks ranging from 1922 to 2011, with a peak in the year 2000s.



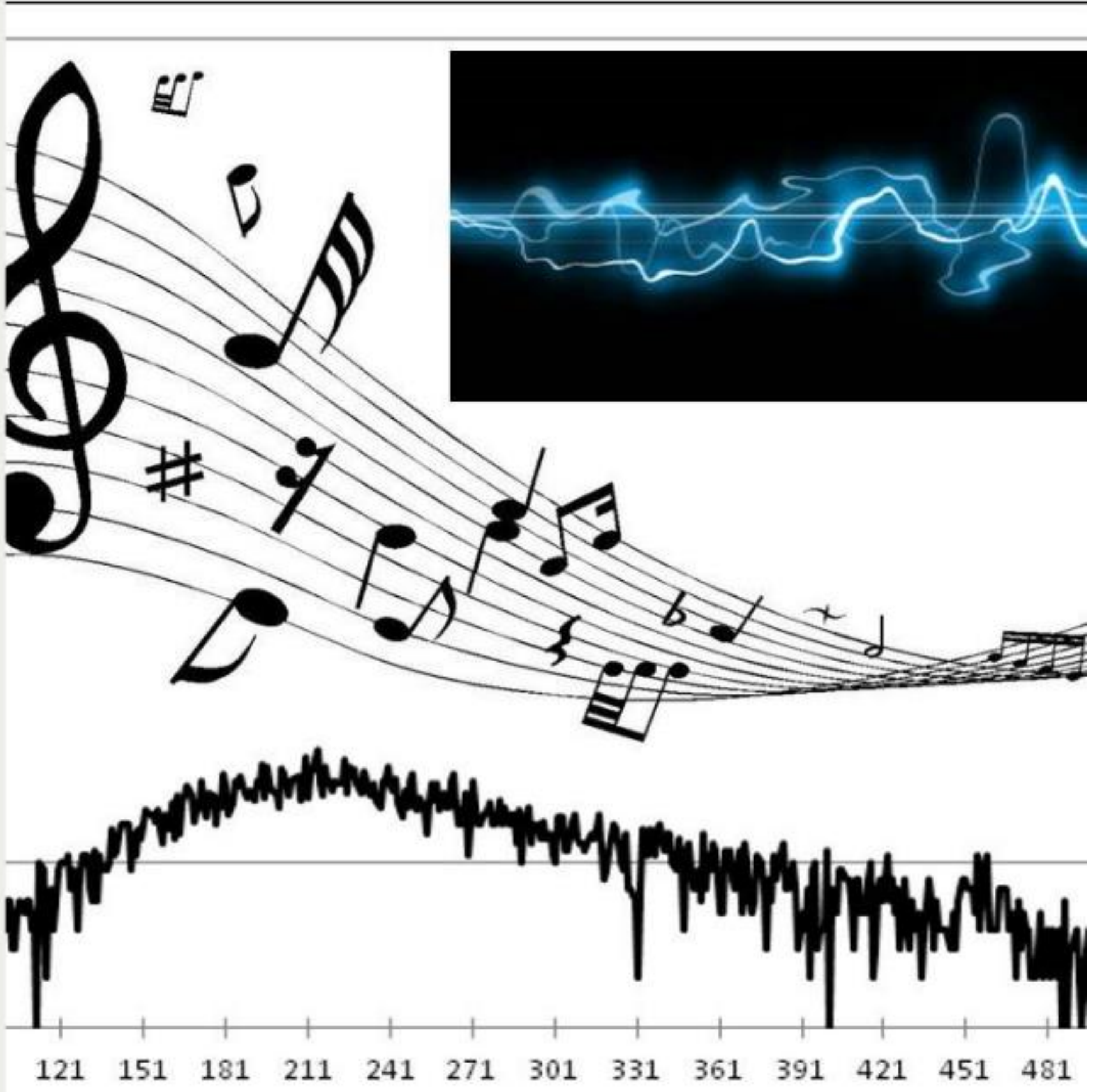
DATASET

- Challenges :

Huge Dataset: The dataset is extensive, with over half a million instances. Running machine learning models on such a massive dataset might pose computational challenges, leading to long training times for each model.

Handling **the variability** in music production styles across different decades.

Extracting **meaningful insights** from the diversity of 91 audio features.



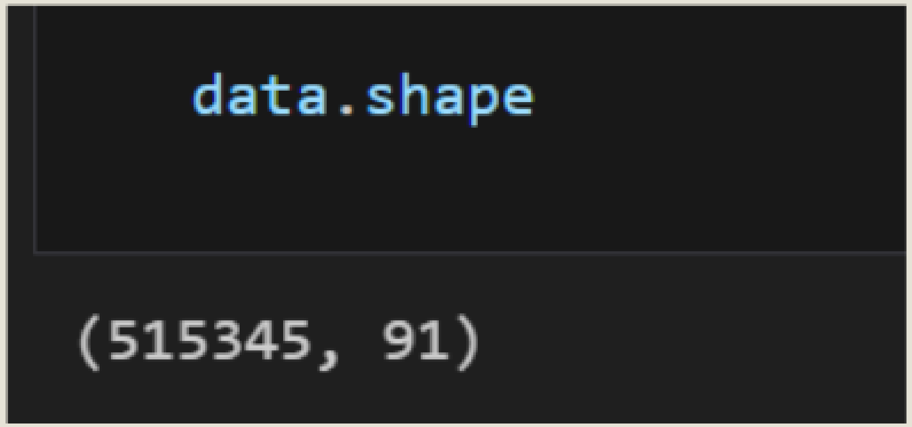
DATA PREPROCESSING

- Removing duplicates
- Removing outliers
- Checking skewness
- Normalisation
- PCA Reduction

REMOVING DUPLICATES

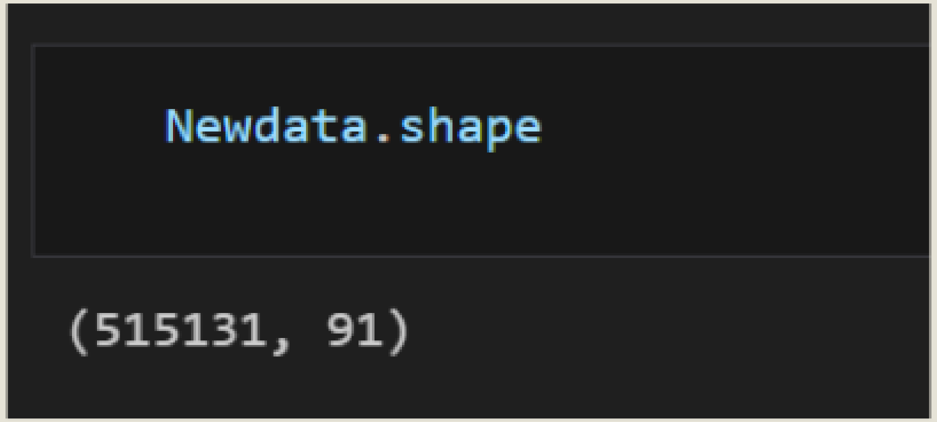
Enhancing Data Integrity

- Initial Data Shape: (515,345, 91)
- After Removing Duplicates: (515,131, 91)



```
data.shape
```

```
(515345, 91)
```



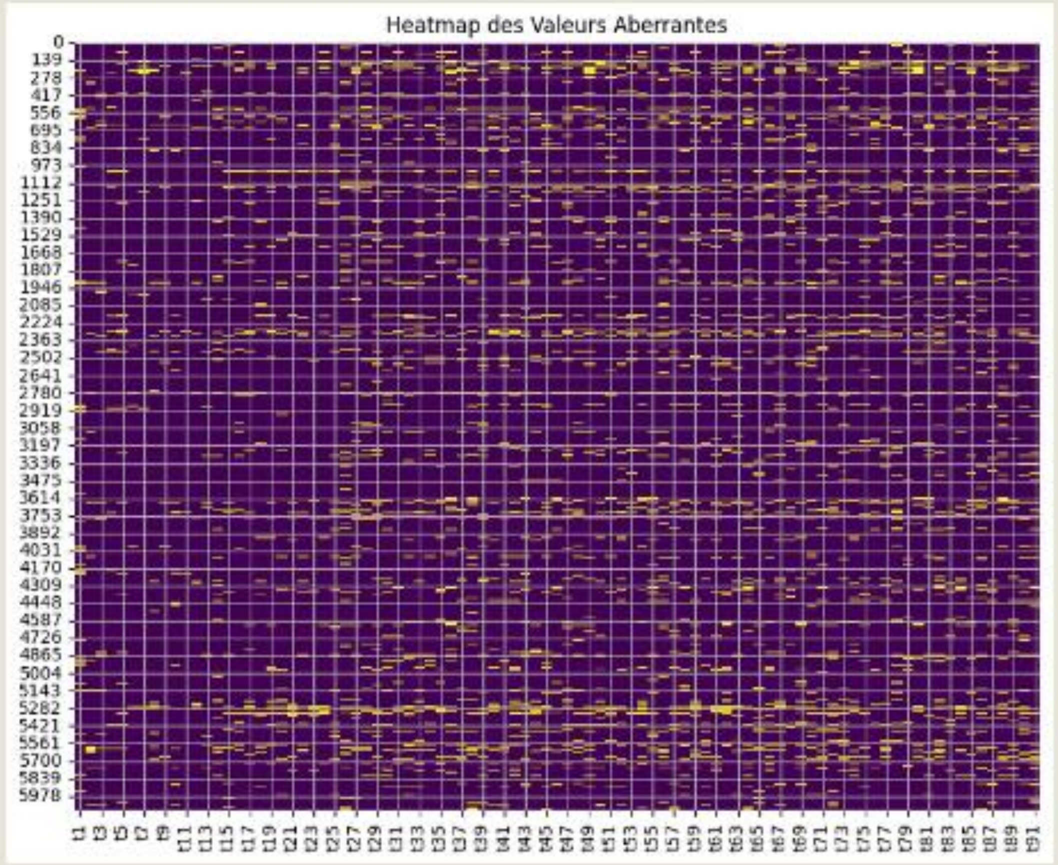
```
Newdata.shape
```

```
(515131, 91)
```

REMOVING OUTLIERS

In the heatmap of outliers, we can observe that they are negligible compared to the rest of the data. This analysis allowed us to determine that the presence of outliers was minimal and only marginally affected the integrity of the overall dataset. Consequently, we decided to remove them.

- Initial Data Shape: (515,345, 91)
- After Removing Outliers: (475,673, 91)



CHEKING SKEWNESS

Assessing Data Distribution

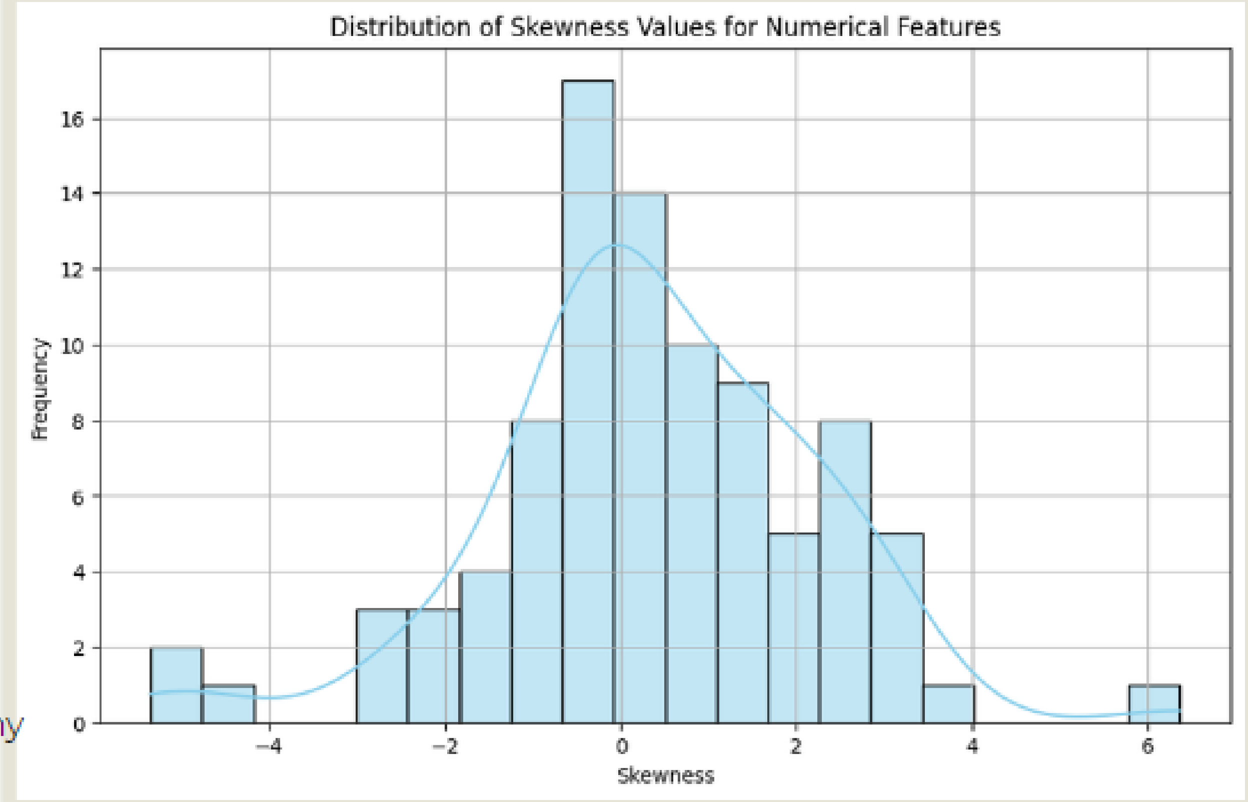
- **Peak at Skewness=0:**

The peak at skewness=0 indicates that a substantial number of your numerical features have a symmetric distribution.

- **Higher Frequency around Skewness Close to 0:**

The higher frequency of features with skewness close to 0 suggests that many features exhibit a near-normal distribution. A skewness value close to 0 implies minimal skewness, and the distribution is relatively balanced.

Having features with skewness close to 0 is favorable. It indicates that the assumptions of normality are reasonably met, which can enhance the reliability of the results obtained from prediction models.

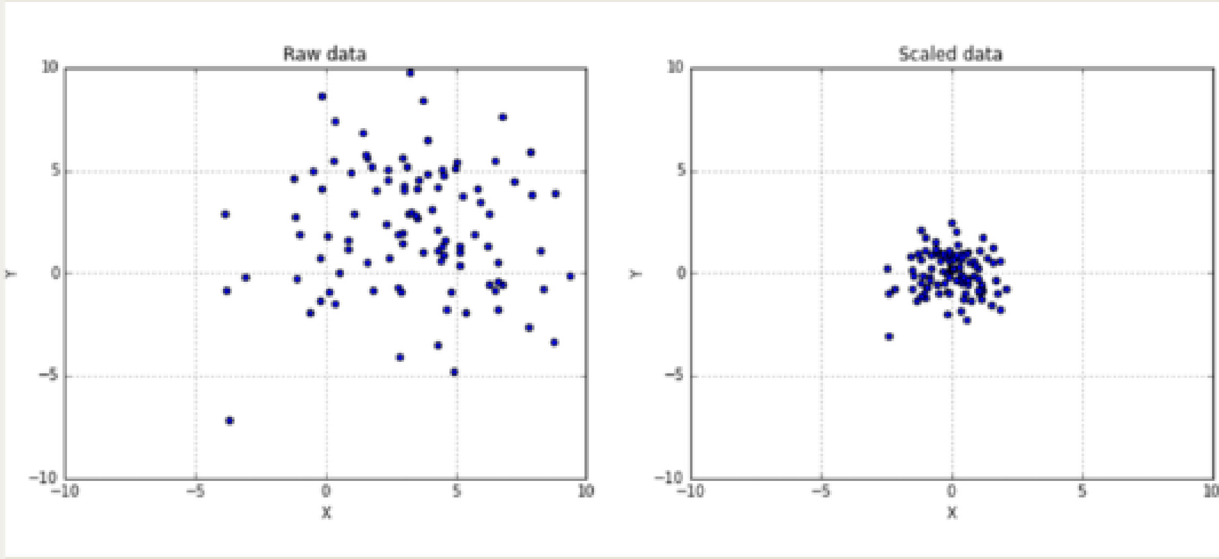


The plot indicates that a substantial portion of the features have distributions close to normal

FEATURE NORMALIZATION

Ensuring Consistent Feature Scales

- Used MinMaxScaler from scikit-learn to scale features between 0 and 1."
- "Fit the scaler on the training set and applied the same transformation to both the training and test sets

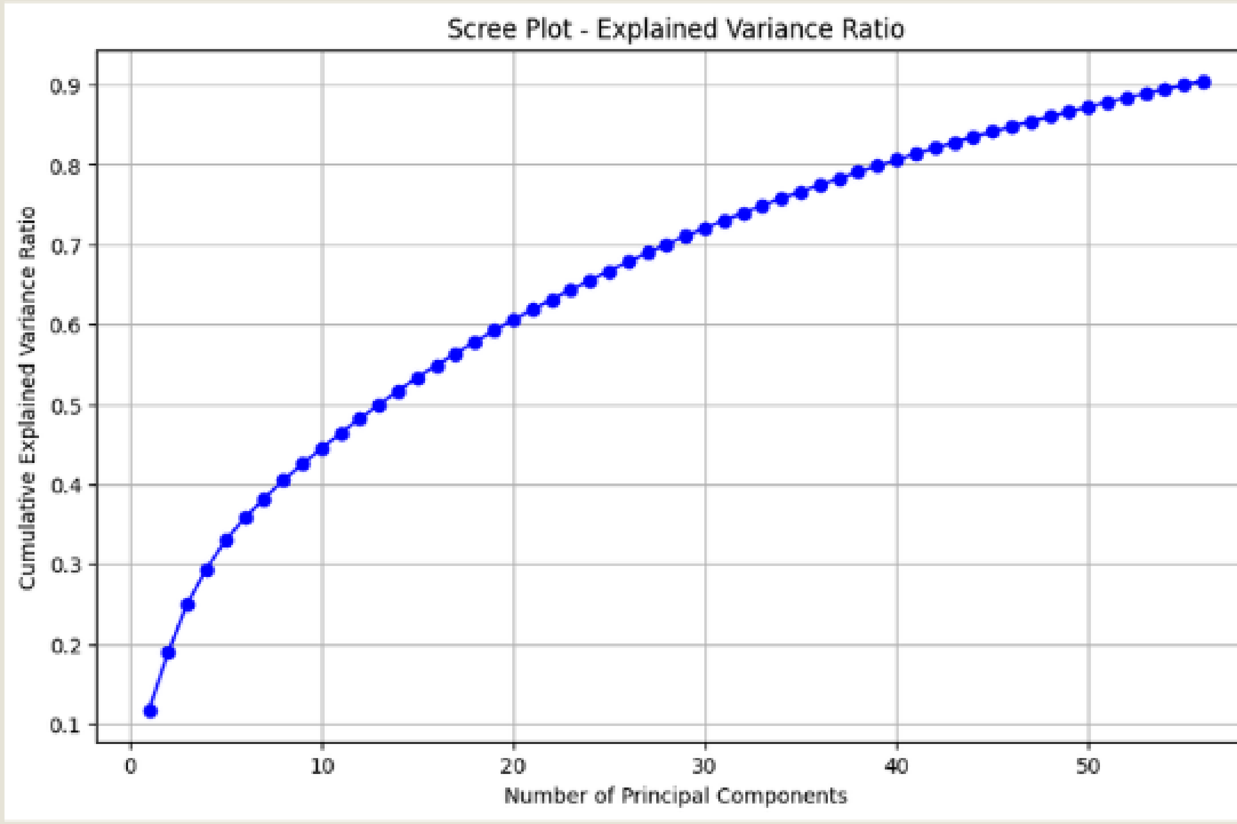


PCA REDUCTION

Dimensionality Reduction for Improved Efficiency

- Working with 90 features can be computationally expensive and may introduce noise
- Original Features: 90
- Implemented PCA to reduce dimensionality and retain 90% of the variance.
- PCA Components Selected: 56 (Preserving 90% Variance)

Reducing features from 90 to 56 through PCA allows for a more efficient and focused representation of the data.



MODELING

- Neural network models
- Linear regression
- Logistic regression
- Random Forest
- Decision Tree

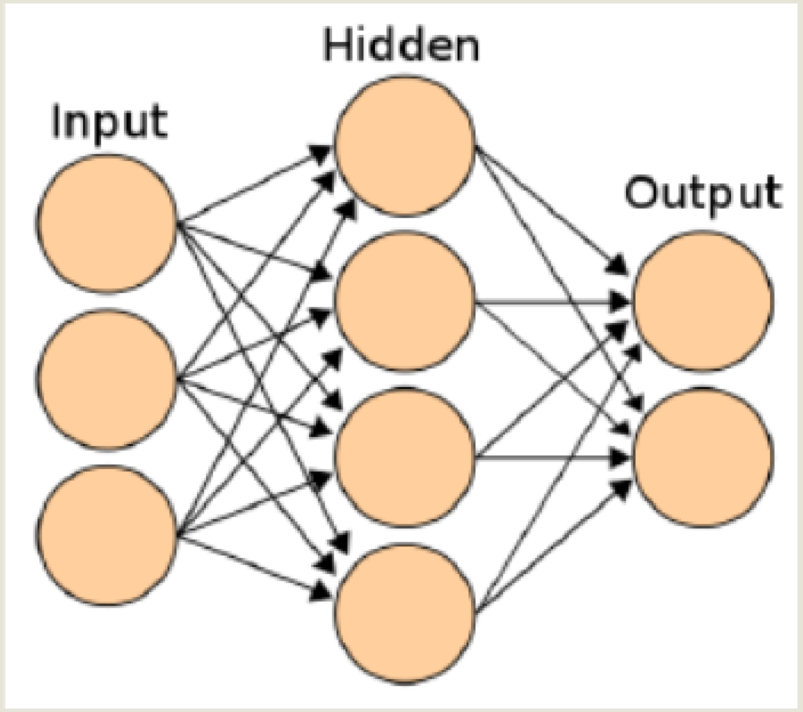
NEURAL NETWORK

Model 1:

- Sequential Model with Three Layers
- **Architecture:**
  "Input Layer: 55 Neurons"
  "Hidden Layer 1: 110 Neurons"
  "Output Layer: 90 Neurons (Softmax Activation)"

- **Activation Functions:**
  "Hidden Layers: Default (usually Rectified Linear Unit - ReLU)"
  "Output Layer: Softmax (for multi-class classification)"

- **Training Details:**
  "Epochs: 5"
  "Batch Size: 64"



NEURAL NETWORK

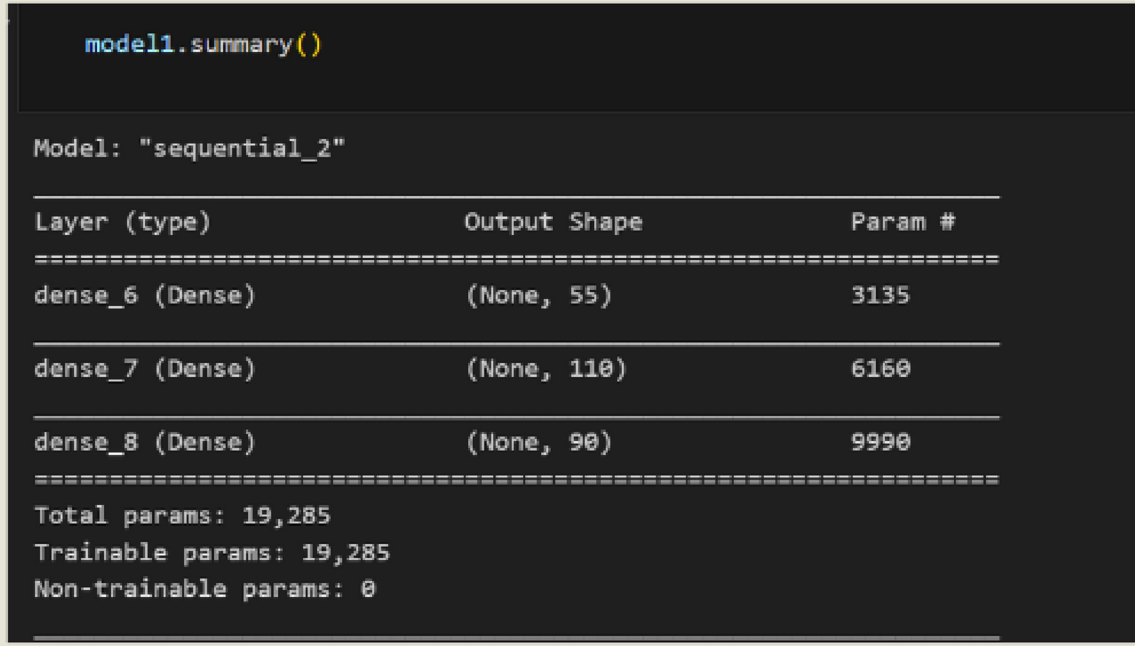
Model 1 evaluation:

The output 9.35 is the mean absolute error (MAE) between the predicted values and the actual values of the target variable for the test set.

- **Interpretation:**

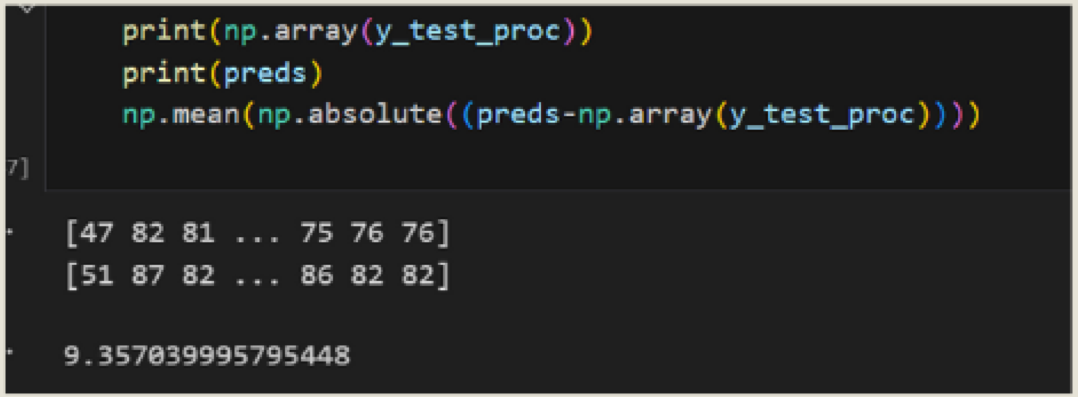
The MAE value of approximately 9.36 suggests that, on average, the absolute difference between your model's predictions and the actual values on the test set is around 9.36 years.

Model's predictions for the year are off by approximately 9.36 years when compared to the actual values in the test set.



```
model1.summary()

Model: "sequential_2"
_________________________________________________________________
Layer (type)                Output Shape              Param #
=================================================================
dense_6 (Dense)             (None, 55)                3135
_________________________________________________________________
dense_7 (Dense)             (None, 110)               6160
_________________________________________________________________
dense_8 (Dense)             (None, 90)                9990
=================================================================
Total params: 19,285
Trainable params: 19,285
Non-trainable params: 0
_________________________________________________________________
```



```
print(np.array(y_test_proc))
print(preds)
np.mean(np.absolute((preds-np.array(y_test_proc))))

[47 82 81 ... 75 76 76]
[51 87 82 ... 86 82 82]

9.357039995795448
```

NEURAL NETWORK

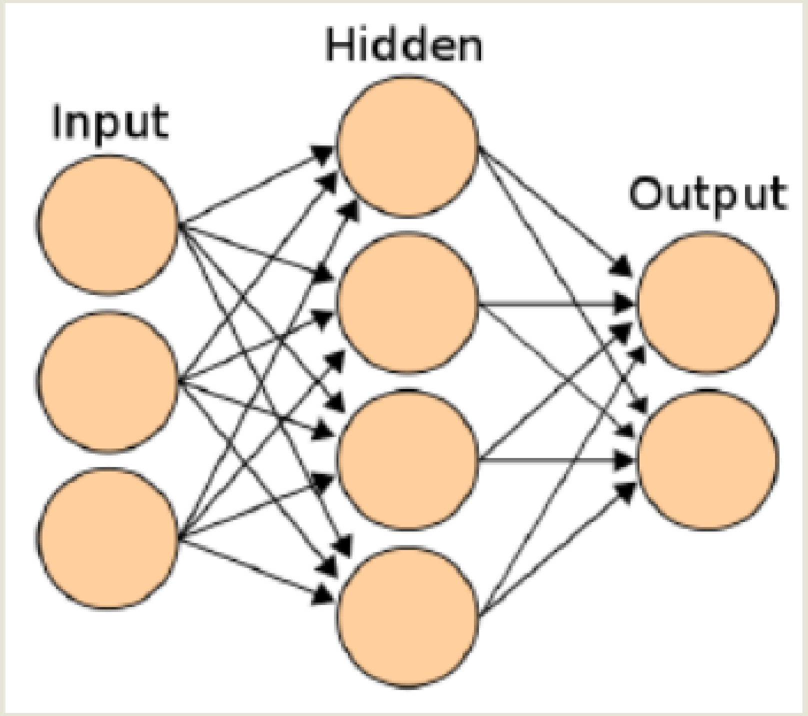
Model 2:

Architecture:

- **Input Layer:**
  - Number of Neurons: 56
  - Activation Function: Rectified Linear Unit (ReLU)
- **Output Layer:**
  - Number of Neurons: 1 (Regression Task)
  - Activation Function: None (Linear Activation)

Compilation:

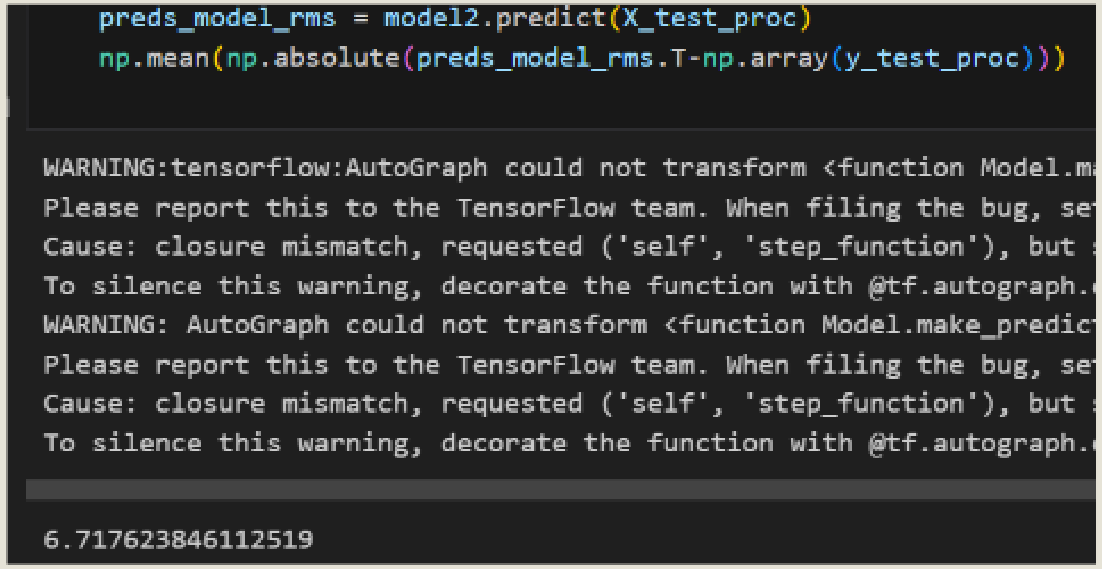
- **Optimizer:**
  - Adam
- **Loss Function:**
  - Mean Squared Error (MSE)
- **Metrics:**
  - Mean Absolute Error (MAE)
- **Training Details**:
  Epochs: 10"
  Batch Size: 64"



NEURAL NETWORK

Model 2 evaluation:

The mean absolute error (MAE) of approximately 6.72 indicates that, on average, your Model 2's predictions for the year are off by approximately 6.72 years when compared to the actual values on the test set.



```
    preds_model_rms = model2.predict(X_test_proc)
    np.mean(np.absolute(preds_model_rms.T-np.array(y_test_proc)))
```

```
WARNING:tensorflow:AutoGraph could not transform <function Model.ma
Please report this to the TensorFlow team. When filing the bug, se
Cause: closure mismatch, requested ('self', 'step_function'), but 
To silence this warning, decorate the function with @tf.autograph.
WARNING: AutoGraph could not transform <function Model.make_predic
Please report this to the TensorFlow team. When filing the bug, se
Cause: closure mismatch, requested ('self', 'step_function'), but 
To silence this warning, decorate the function with @tf.autograph.
```

```
6.717623846112519
```

NEURAL NETWORK

Model 3:

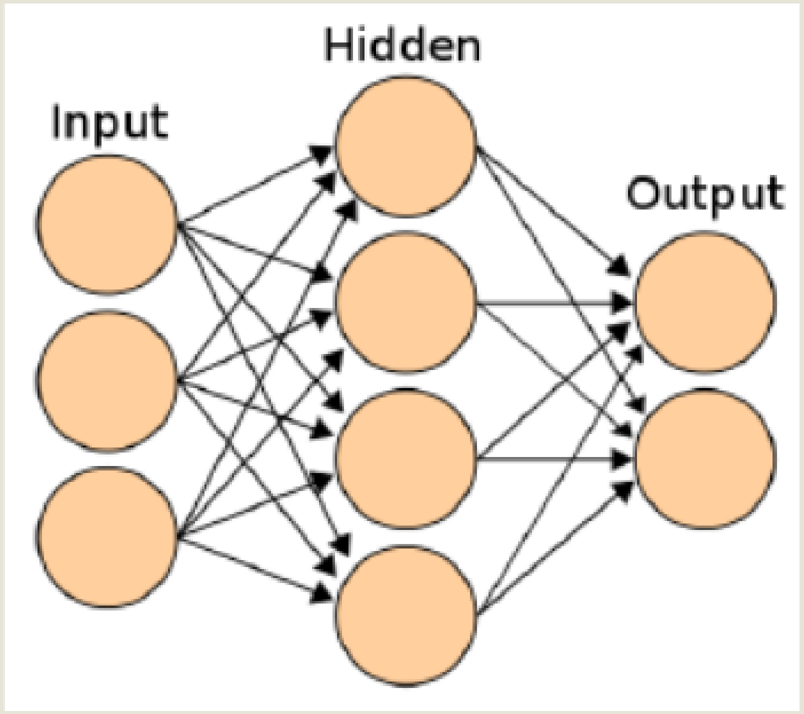
- **Architecture:**

1. Input Layer:
   - Number of Neurons: 55
   - Activation Function: Rectified Linear Unit (ReLU)
   - Input Shape: (56,)
2. Hidden Layer 1:
   - Number of Neurons: 110
   - Activation Function: Rectified Linear Unit (ReLU)
3. Dropout Layer:
   - Rate: 0.2 (20% dropout rate)
   - Introduces regularization to prevent overfitting by randomly setting 20% of the input units to 0 at each update during training.
4. Output Layer:
   - Number of Neurons: 1
   - Activation Function: None (linear activation for regression)

- **Compilation:**
- Optimizer:
  - Adam
- Loss Function:
  - Mean Squared Error (MSE)
- Metrics:
  - Mean Absolute Error (MAE)

Training Details:

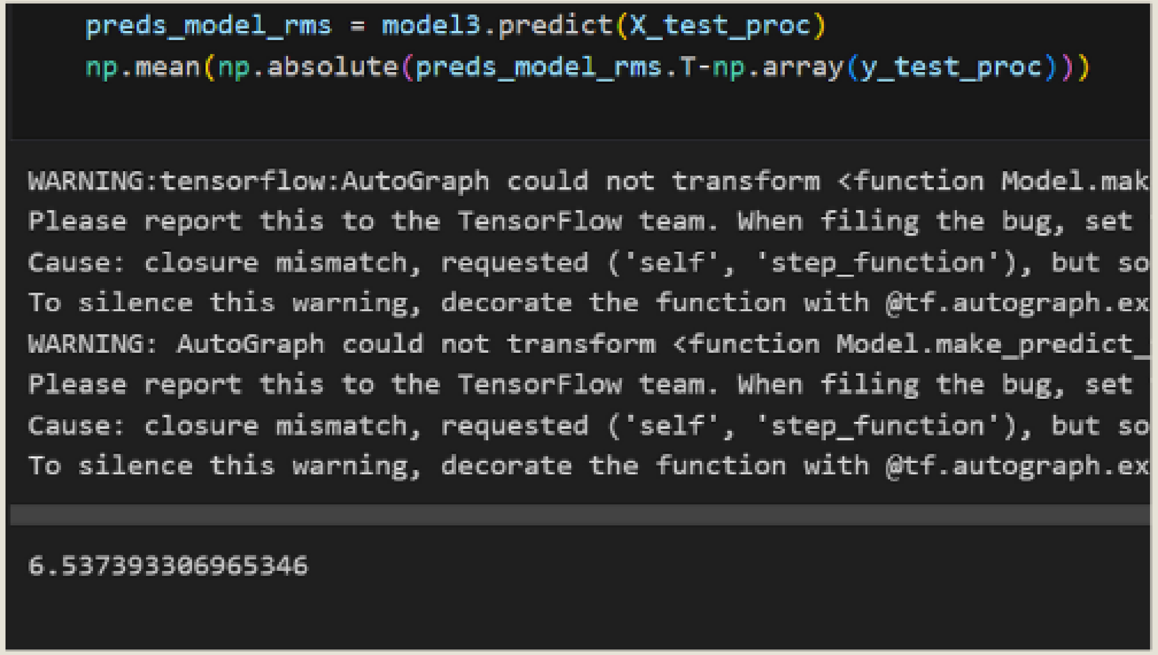
- Epochs:
  - 10
- Batch Size:
  - 128



NEURAL NETWORK

Model 3 evaluation:

The mean absolute error (MAE) of approximately 6.53 indicates that, on average, your Model 2's predictions for the year are off by approximately 6.53 years when compared to the actual values on the test set.



```
preds_model_rms = model3.predict(X_test_proc)
np.mean(np.absolute(preds_model_rms.T-np.array(y_test_proc)))
```

```
WARNING:tensorflow:AutoGraph could not transform <function Model.mak
Please report this to the TensorFlow team. When filing the bug, set
Cause: closure mismatch, requested ('self', 'step_function'), but so
To silence this warning, decorate the function with @tf.autograph.ex
WARNING: AutoGraph could not transform <function Model.make_predict_
Please report this to the TensorFlow team. When filing the bug, set
Cause: closure mismatch, requested ('self', 'step_function'), but so
To silence this warning, decorate the function with @tf.autograph.ex
```

```
6.537393306965346
```

LINEAR REGRESSION

How it Works:

- Logistic Regression models the relationship between the input features and the probability of a song belonging to a specific release year category.

Probability Output:

- Instead of predicting a specific year directly, the model outputs probabilities for each possible release year.

Decision Boundary:

- A decision boundary is established to classify songs into specific release year categories based on the predicted probabilities.

LOGISTIC REGRESSION

How it Works:

- Linear Regression models the relationship between input features and the predicted release year through a linear equation.

Continuous Prediction:

- If predicting a continuous release year, the model provides a direct numerical output.

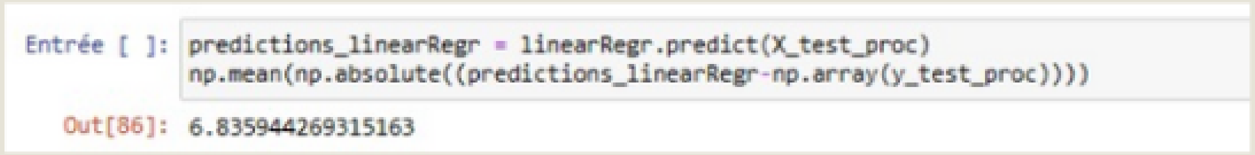
Coefficients Interpretation:

- Coefficients in the linear equation represent the contribution of each feature to the predicted release year.

Limitations:

- Assumes a linear relationship, which may not capture complex patterns in the data.

LINEAR REGRESSION

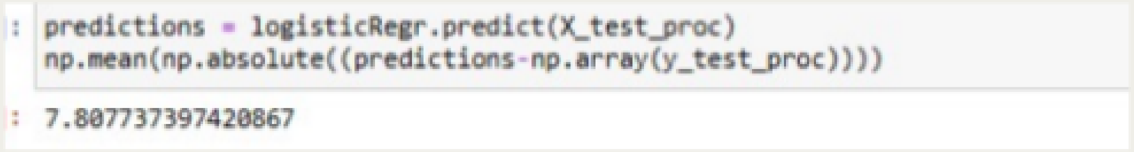


```
Entrée [ ]: predictions_linearRegr = linearRegr.predict(X_test_proc)
            np.mean(np.absolute((predictions_linearRegr-np.array(y_test_proc))))

   Out[86]: 6.835944269315163
```

- The linear regression model, on average, predicts the release year with an absolute error of approximately 6.83 years. This implies that the predictions deviate from the true release years by around 6.83 years.

LOGISTIC REGRESSION



```
: predictions = logisticRegr.predict(X_test_proc)
  np.mean(np.absolute((predictions-np.array(y_test_proc))))

: 7.807737397420867
```

- The logistic regression model, on average, predicts the release year with an absolute error of approximately 7.80 years. This indicates an average deviation of around 7.80 years between the predicted and actual release years.

LINEAR REGRESSION

LOGISTIC REGRESSION

Comparative Analysis:

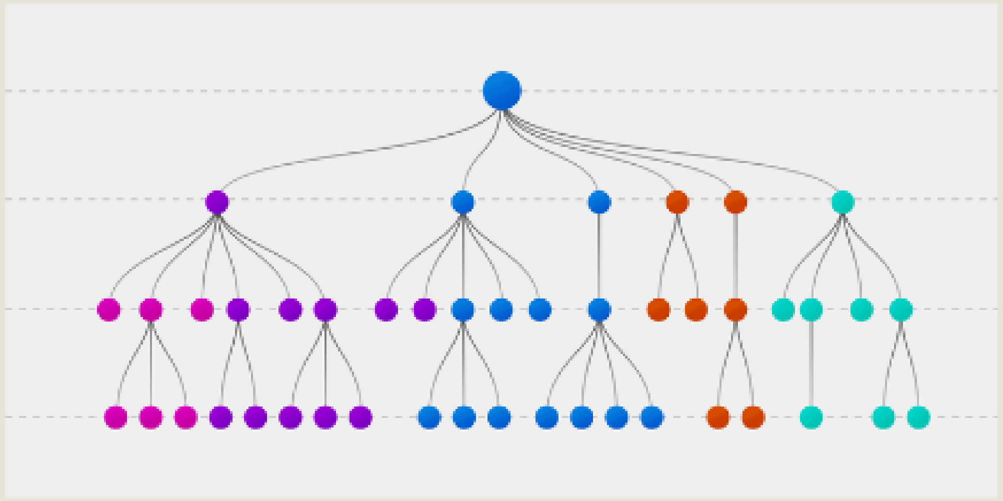
- The linear regression model performs slightly better than the logistic regression model in terms of MAE. A lower MAE is desirable as it indicates smaller prediction errors.

Reasonable Performance:

- Considering that the target variable spans different decades, the achieved MAE values (6.83 and 7.80) may be considered reasonable. The task of predicting release years, especially across diverse decades, is inherently challenging due to the complexity of musical trends and evolution.

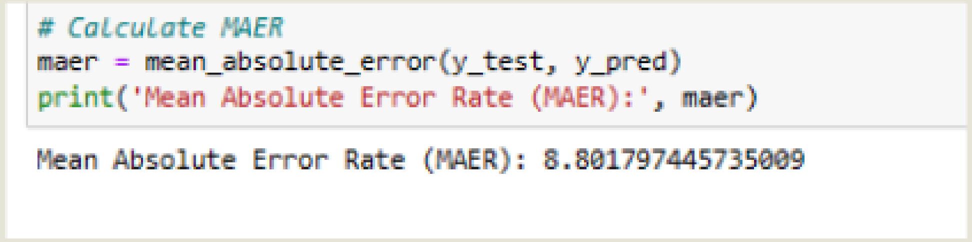
DECISION TREE

In the context of predicting song release years, a decision tree model is a machine learning algorithm that learns a hierarchical series of decision rules based on audio features or relevant attributes of songs. These rules, represented in a tree-like structure, guide the model in making predictions about the likely release year of a song.



Decision Tree Model (MAE: 8.80):

The decision tree model, on average, has a higher absolute error of approximately 8.80 years. This means that the decision tree's predictions deviate, on average, by around 8.80 years from the true release years.



```
# Calculate MAER
maer = mean_absolute_error(y_test, y_pred)
print('Mean Absolute Error Rate (MAER):', maer)
```

```
Mean Absolute Error Rate (MAER): 8.801797445735009
```

RANDOM FOREST CLASSIFIER 1

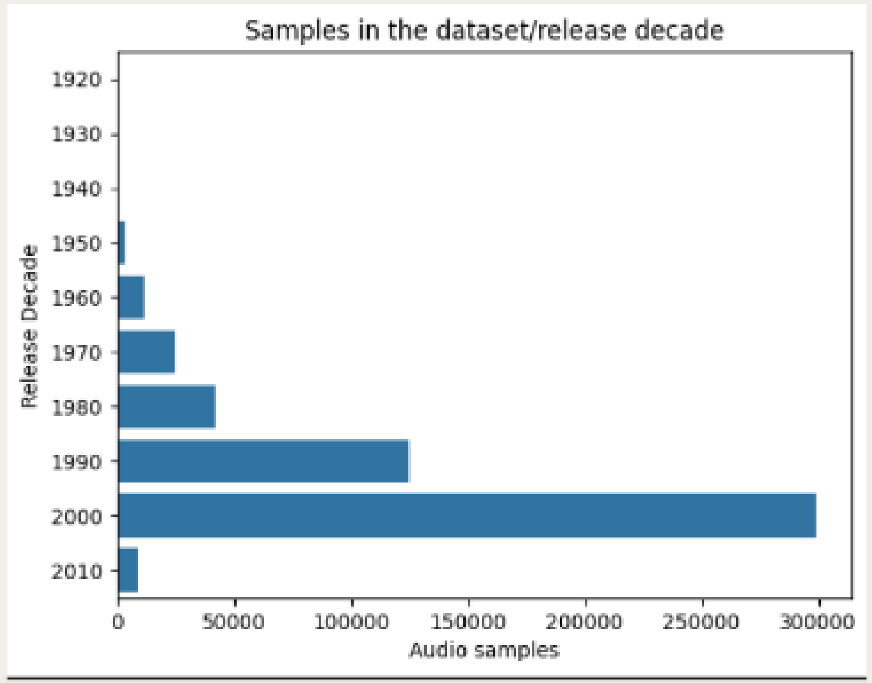
First attempt :

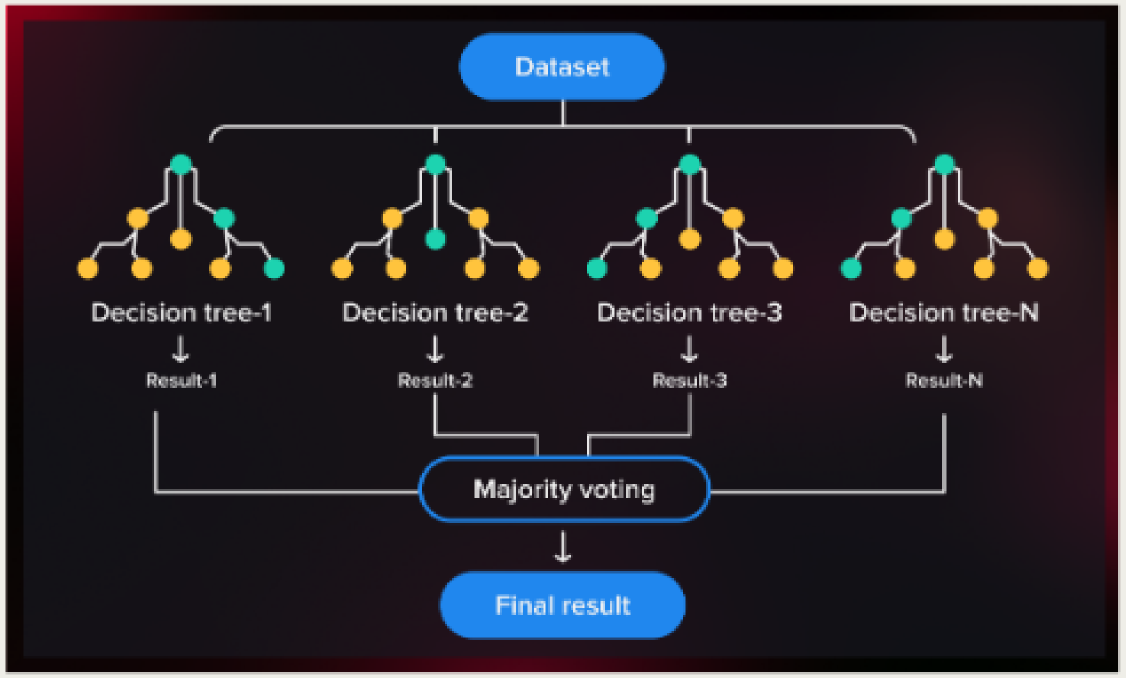
For time and ressources reasons, we decided to target decade of the release instead of the exact year. We modified the the labels so it was shown as decade.

Parameters :

100 decision trees
criterion = gini
no max depth

Then, we applied a Radom Forest Classifier with auto parameters. The resulst are quite satisfying with an accuracy of 60% and a mean absolute error of 6.48.





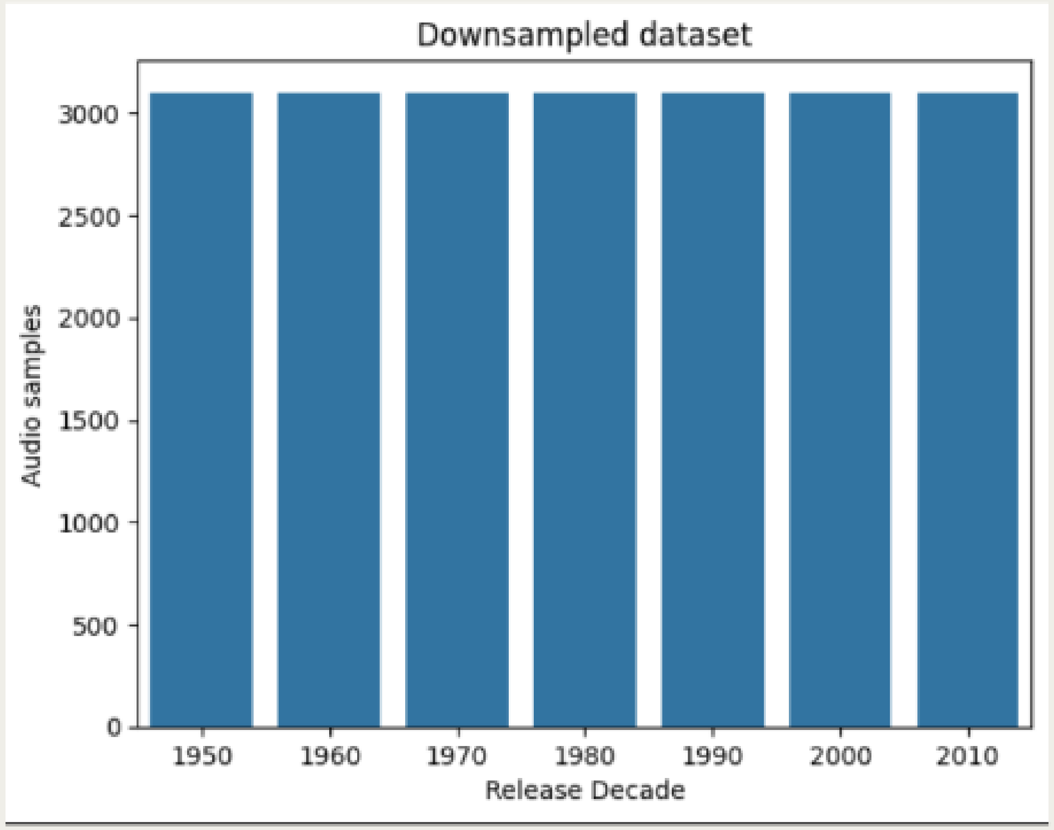
RANDOM FOREST CLASSIFIER 2

Second attempt :

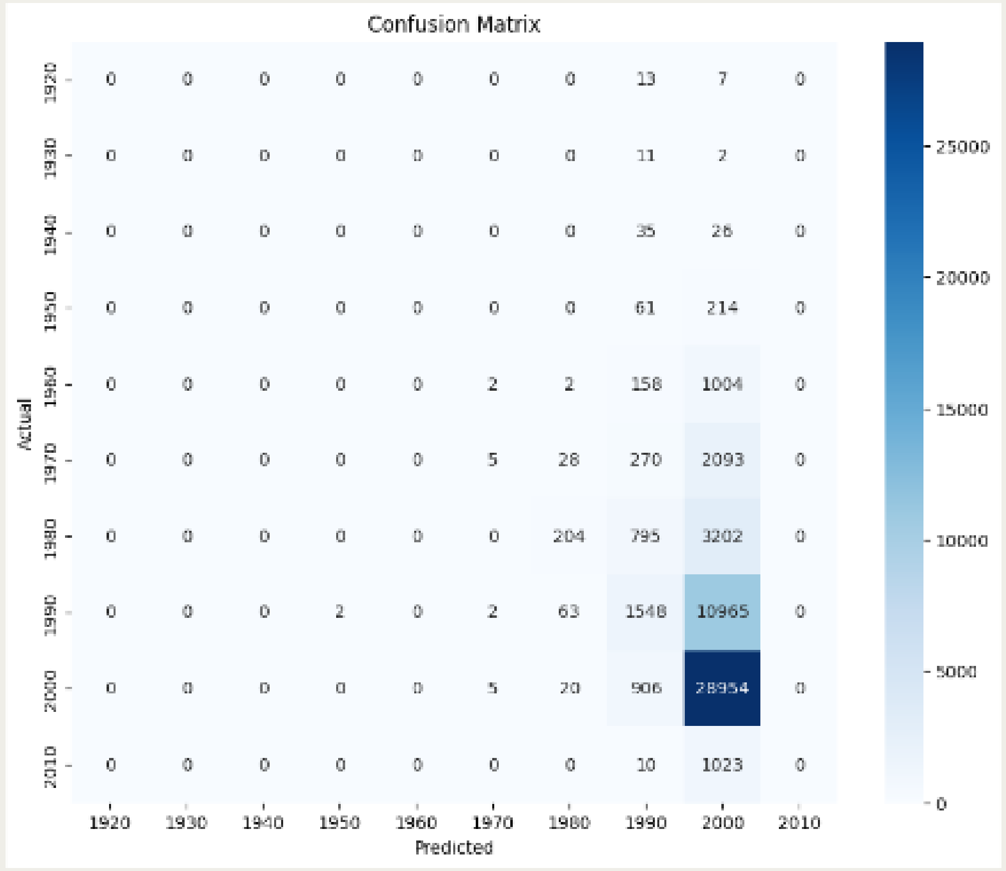
In our second attempt to use Random Forest Classifier, we tried to an even repartition of the decades because some decades are under represented. In the new dataset, each decade from 1950 to 2010 have the same number of datapoints (3102).

Parameters :
100 decision trees
criterion = gini
no max depth

Unfortunately, this method wasn't very effective and we got an accuracy of 13% and a MAE of 17 years.



RFC Confusion Matrix



Actual \ Predicted	1920	1930	1940	1950	1960	1970	1980	1990	2000	2010
1920	0	0	0	0	0	0	0	13	7	0
1930	0	0	0	0	0	0	0	11	2	0
1940	0	0	0	0	0	0	0	35	26	0
1950	0	0	0	0	0	0	0	61	214	0
1960	0	0	0	0	0	2	2	158	1004	0
1970	0	0	0	0	0	5	28	270	2093	0
1980	0	0	0	0	0	0	204	795	3202	0
1990	0	0	0	2	0	2	63	1548	10965	0
2000	0	0	0	0	0	5	20	906	28954	0
2010	0	0	0	0	0	0	0	10	1023	0

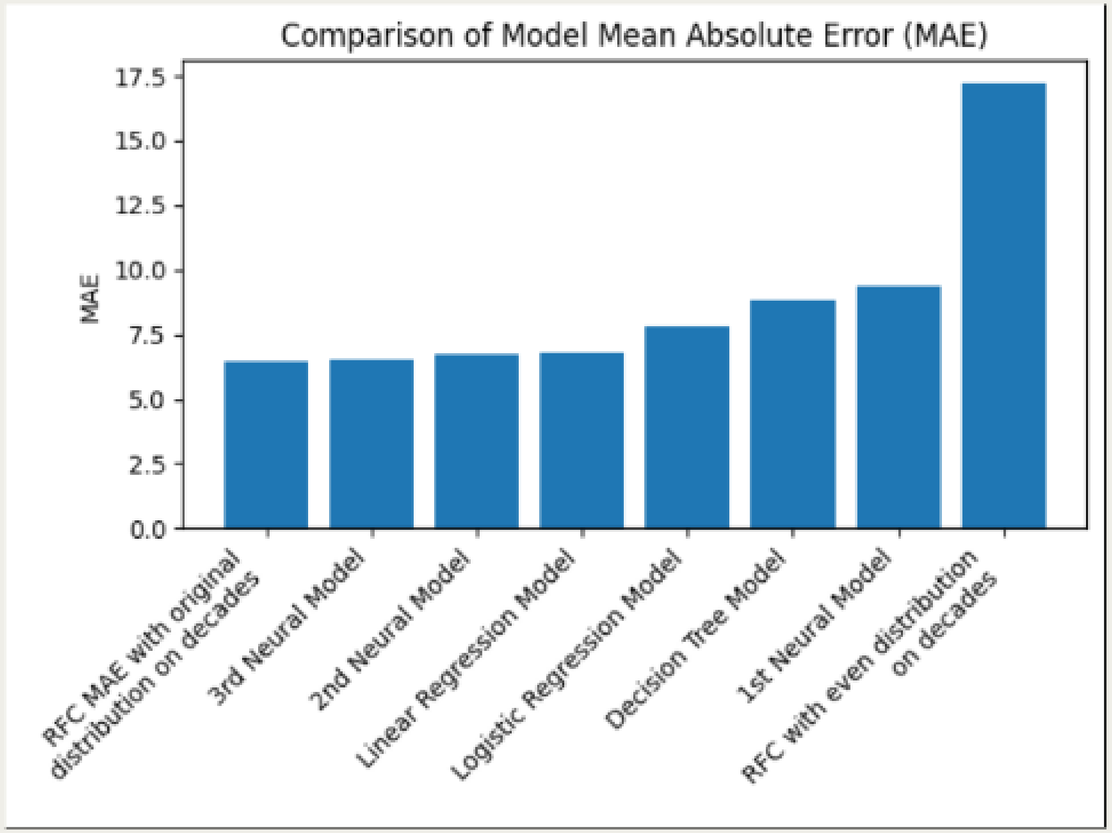
RFC Model 1 Confusion Matrix

We can see the problem here is the overrepresentated 2000 decade.

Most of the song are in 2000 decade and the model have a tendency to put the label 2000 even when it’s not true.

Therefore, the model is accurate not because it’s well trained but because there is more chance to have a good answer in putting 2000 label on a datapoint.

Model Comparison using Mean Absolute Error



Here is a comparison of the efficiency of the models using the Mean Absolute error

We can see that the RFC with original data, the 3rd and 2nd Neural Model and the Linear Regression are the most accurate in our case.