Spotify

SPOTIFY TRENDS

Present by Nilufar Iangiboeva

https://www.kaggle.com/datasets/dhruvildave/spotify-charts

902431@stud.unive.it

AGENDA

1 Problem Statement

2 Data

3 Descriptive statistics

4 Feature engineering

5 Supervised Learning Method

6 Neural Network

7 Conclusion

1. PROBLEM STATEMENT

Objective: Understand the factors that influence a song's inclusion in Spotify's "Top 200" and "Viral 50" charts and provide actionable insights for achieving chart success.

Key Goals:

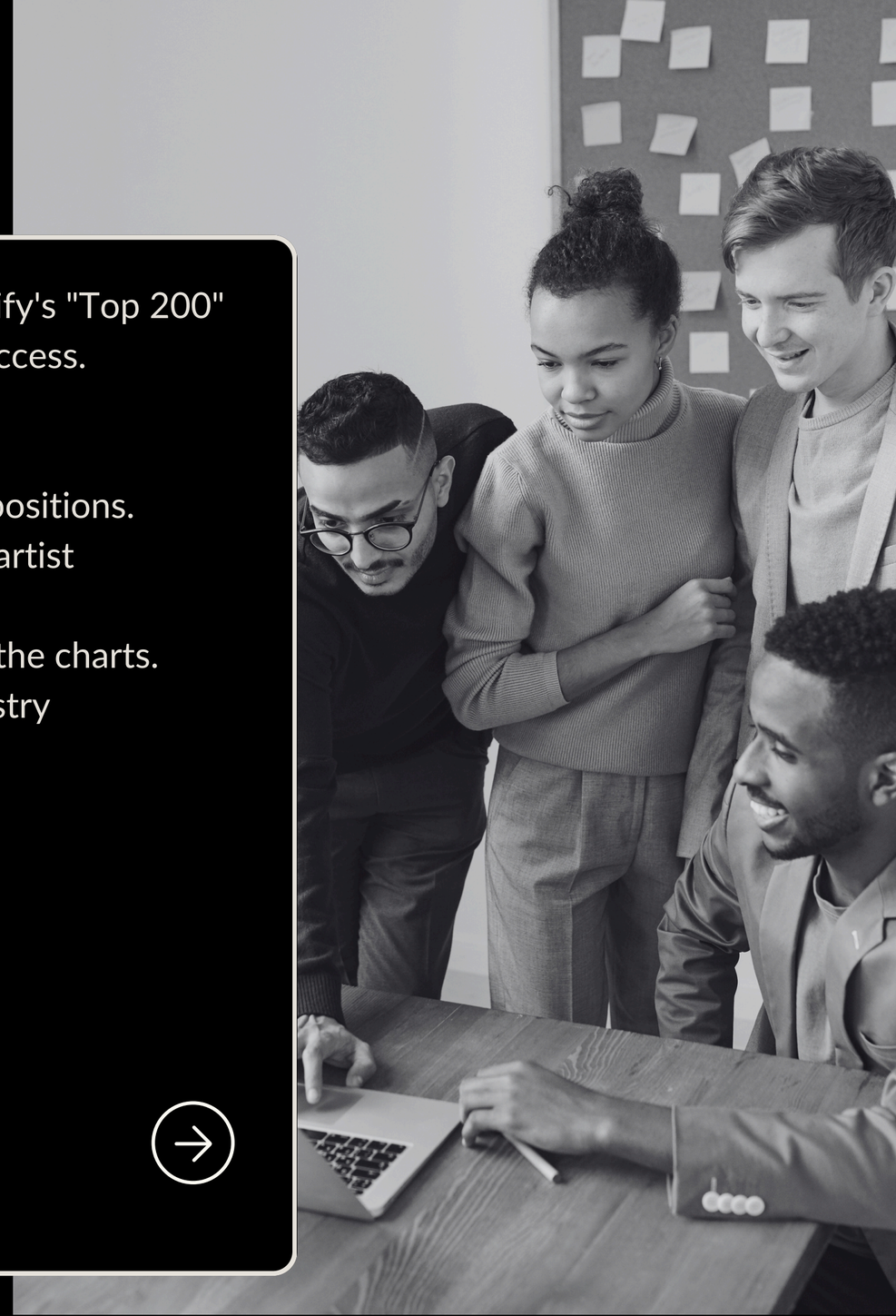
1. Analyze Chart Composition: Identify key factors that determine chart positions.
2. Evaluate Feature Impact: Assess the influence of attributes like genre, artist popularity, and promotions.
3. Develop Predictive Models: Forecast the likelihood of a song entering the charts.
4. Provide Strategic Insights: Offer recommendations for artists and industry stakeholders.

Challenges:

- Complex influences on chart positions.
- Managing diverse and voluminous data.
- Rapidly changing music trends.

Expected Outcomes:

- A clear understanding of chart determinants.
- Predictive models for chart success.
- Insights on feature importance.
- Practical strategies for charting success.



2. DATA

Description

This is a complete dataset of all the "Top 200" and "Viral 50" charts published globally by Spotify. Spotify publishes a new chart every 2-3 days. This is its entire collection since January 1, 2017.

https://www.kaggle.com/datasets/dhruvildave/spotify-charts

Size

26173514 observations
9 columns

Info

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 26173514 entries, 0 to 26173513
Data columns (total 9 columns):
 #   Column   Dtype
---  ------   -----
 0   title    object
 1   rank     int64
 2   date     object
 3   artist   object
 4   url      object
 5   region   object
 6   chart    object
 7   trend    object
 8   streams  float64
dtypes: float64(1), int64(1), object(7)
memory usage: 1.8+ GB
```

PREPROCESSING

Analysis

Before building our machine learning system we must get our data into the optimal shape for our learning algorithm.

Handling NA

The most popular song of this period by this Artist was NA, so I fill in NA

To simplify the calculation I want to remove the missing values just for the project, The value of streams is NULL when the chart column is "viral50".

Removed 5851592 rows, which is **22.36%** of the dataset.

Reducing data for fast working

Random sampling ->> 1 000 000 observations left

3. DESCRIPTIVE STATISTICS

SONGS & ARTISTS

The sample contains 1000000 songs and 23000 artists

REGIONS

The dataset contains 69 regions

CHARTS

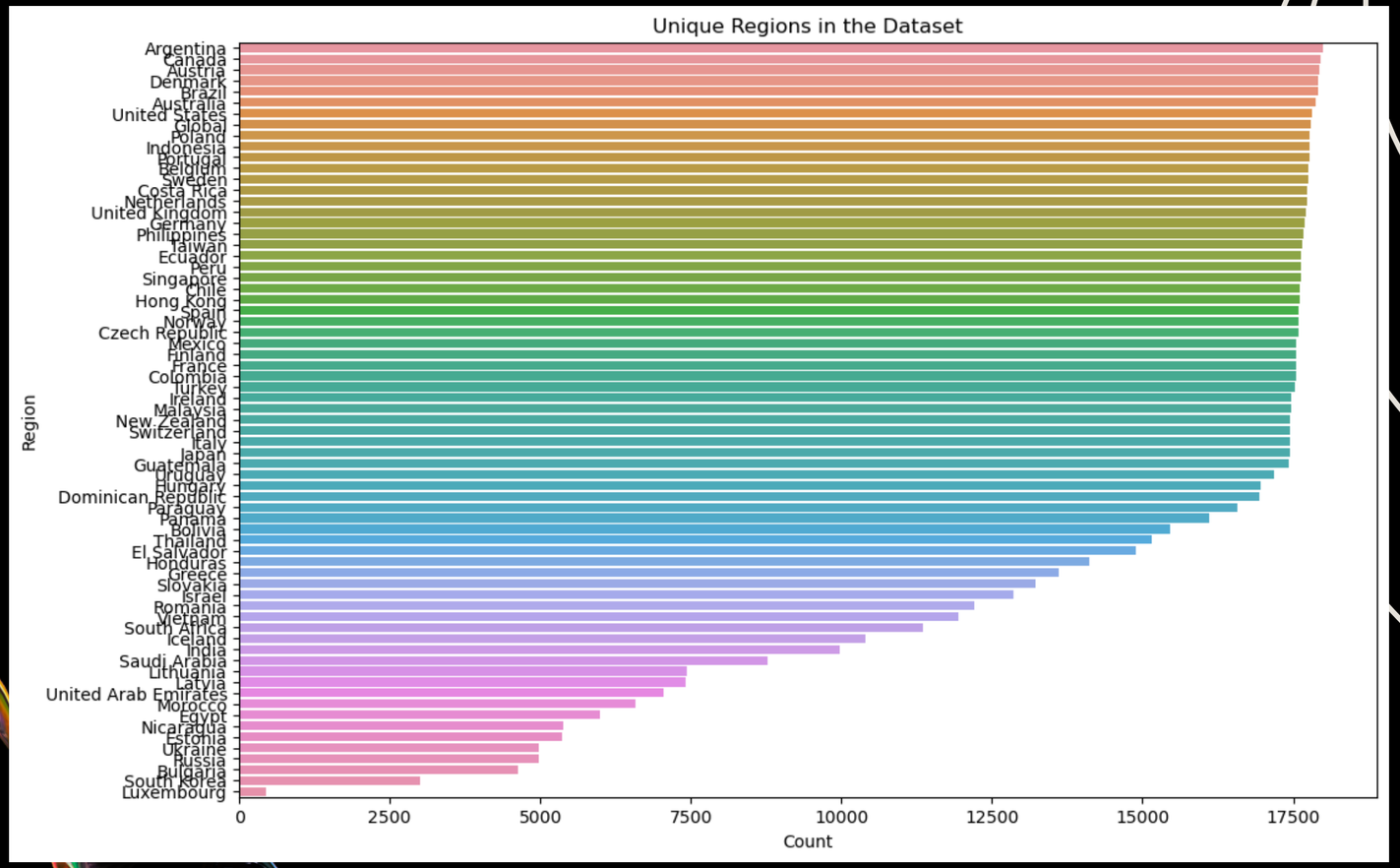
The dataset contains 1 chart

TRENDS

The dataset contains 4 trends:

MOVE_DOWN	MOVE_UP
NEW_ENTRY	SAME_POSITION

REGION DISTRIBUTION



The most listening regions are: Argentina, Canada, Austria, Denmark, Brazil

The medium listeners: Czech Republic, Mexico, Finland, Colombia, Turkey

Less listeners: Estonia, Ukraine, Russia, Bulgaria, Luxemburg

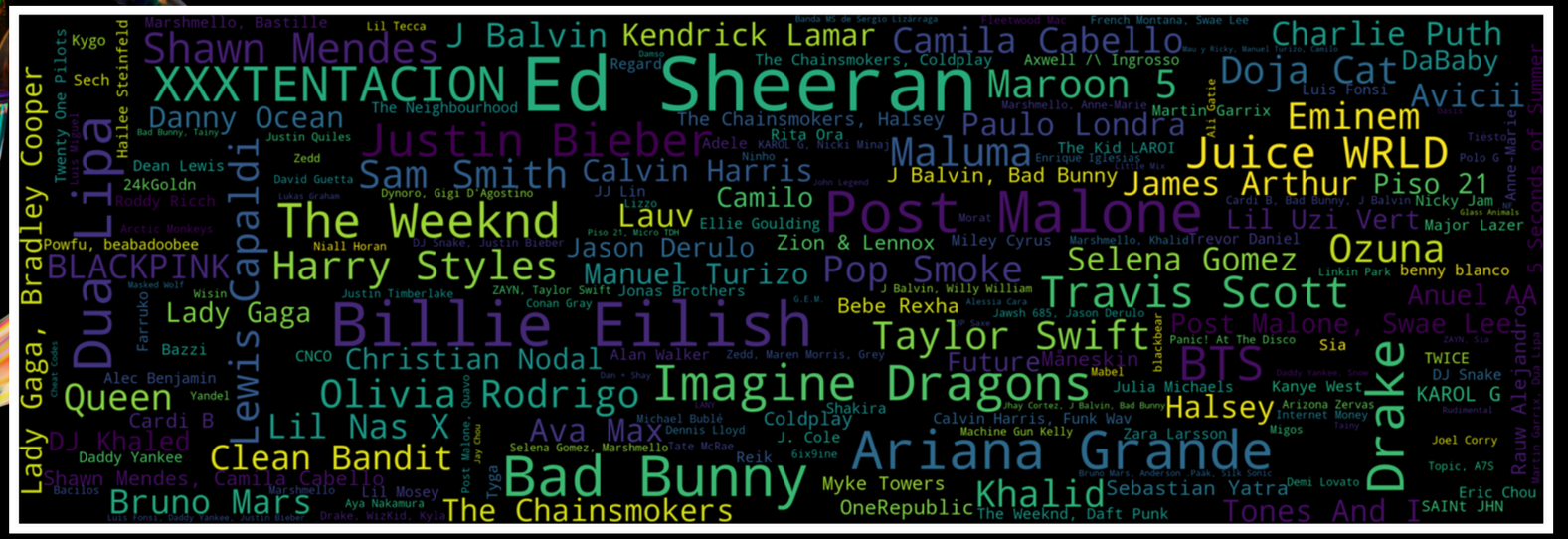
ALL REGIONS AND THEIR TOP SONG

	Region	Top Song
0	Japan	Shape of You
1	Colombia	Rojo
2	Taiwan	刻在我心底的名字 (Your Name Engraved Herein) – 電影<刻在你心...
3	Indonesia	Havana
4	Spain	Despacito (Featuring Daddy Yankee)
..	...	...
64	Bulgaria	Astronaut In The Ocean
65	Estonia	tmt
66	Ukraine	Снова я напиваюсь
67	Luxembourg	Shape of You
68	South Korea	My Universe

ALL REGIONS AND THEIR TOP ARTIST

	Region	Top Artist
0	Japan	Ed Sheeran
1	Colombia	J Balvin
2	Taiwan	Crowd Lu
3	Indonesia	Maroon 5
4	Spain	Luis Fonsi
..	...	...
64	Bulgaria	Lil Nas X
65	Estonia	nublu
66	Ukraine	SLAVA MARLOW
67	Luxembourg	Ed Sheeran
68	South Korea	BTS

POPULAR ARTISTS



Top 5 most streamed songs

Blinding Lights by The Weeknd (280577636.0 streams)
Shape of You by Ed Sheeran (237736544.0 streams)
Señorita by Shawn Mendes, Camila Cabello (233146792.0 streams)
Dance Monkey by Tones And I (228507650.0 streams)
Someone You Loved by Lewis Capaldi (208249049.0 streams)

Bottom 5 streamed songs

Supersonic (My Existence) [with Noisia, josh pan & Dylan Brady] by Skrillex (1001.0 streams)
Dumblebeat by TwickBeats (1002.0 streams)
Ir Vērts by Singapūras Satīns (1002.0 streams)
Kalandor by Soulwave (1002.0 streams)
Papi by Maian (1002.0 streams)

4. FEATURE ENGINEERING

I remove irrelevant features, encode our target variable Y, and encode categorical variables using LabelEncoder and One-hot encoding

```
#encode categorical variables
label_encoder_artist = LabelEncoder()
label_encoder_title = LabelEncoder()
spot_cleaned_cls['artist_encoded'] = label_encoder_artist.fit_transform(spot_cleaned_cls['artist'])
spot_cleaned_cls['title_encoded'] = label_encoder_title.fit_transform(spot_cleaned_cls['title'])
spot_cleaned_cls['region_encoded'] = label_encoder_title.fit_transform(spot_cleaned_cls['region'])
spot_cleaned_cls.head()
```

	title	rank	artist	region	trend	streams	year	month	day	season	artist_encoded	title_encoded	region_encoded
9999818	Slow & Easy	75	HIRAIDAI	Japan	MOVE_DOWN	3889.0	2017	8	11	3	8048	33301	33
9126206	Magia (feat. Sebastián Yatra)	191	Andrés Cepeda, Sebastian Yatra	Colombia	NEW_ENTRY	7986.0	2020	4	4	2	1285	22094	9
14787372	她沒在看我	2	E.SO	Taiwan	MOVE_DOWN	43495.0	2020	8	7	3	5750	46035	60
5693904	Hitam Putih	143	Fourtwnty	Indonesia	SAME_POSITION	25421.0	2019	4	25	2	7035	15310	29
22953588	Vida de Rico	53	Camilo	Spain	MOVE_DOWN	115583.0	2021	8	13	3	3458	38799	57

```
#encode our target variable Y
spot_cleaned_cls['trend'].unique()
trend_mapping = {'SAME_POSITION': 1, 'MOVE_UP': 2, 'MOVE_DOWN': 3, 'NEW_ENTRY': 4}
spot_cleaned_cls['trend_encoded'] = spot_cleaned_cls['trend'].map(trend_mapping)

spot_cleaned_cls.head()
```

	title	rank	artist	region	trend	streams	year	month	day	season	artist_encoded	title_encoded	region_encoded	trend_e
9999818	Slow & Easy	75	HIRAIDAI	Japan	MOVE_DOWN	3889.0	2017	8	11	3	8048	33301	33	
9126206	Magia (feat. Sebastián Yatra)	191	Andrés Cepeda, Sebastian Yatra	Colombia	NEW_ENTRY	7986.0	2020	4	4	2	1285	22094	9	
14787372	她沒在看我	2	E.SO	Taiwan	MOVE_DOWN	43495.0	2020	8	7	3	5750	46035	60	
5693904	Hitam Putih	143	Fourtwnty	Indonesia	SAME_POSITION	25421.0	2019	4	25	2	7035	15310	29	
22953588	Vida de Rico	53	Camilo	Spain	MOVE_DOWN	115583.0	2021	8	13	3	3458	38799	57	

```
#remove irrelevant fetures
spot_cleaned_cls=spot_cleaned.drop(['url', 'chart', 'date'], axis=1)
spot_cleaned_cls.head()
```

	title	rank	artist	region	trend	streams	year	month	day	season
9999818	Slow & Easy	75	HIRAIDAI	Japan	MOVE_DOWN	3889.0	2017	8	11	3
9126206	Magia (feat. Sebastián Yatra)	191	Andrés Cepeda, Sebastian Yatra	Colombia	NEW_ENTRY	7986.0	2020	4	4	2
14787372	她沒在看我	2	E.SO	Taiwan	MOVE_DOWN	43495.0	2020	8	7	3
5693904	Hitam Putih	143	Fourtwnty	Indonesia	SAME_POSITION	25421.0	2019	4	25	2
22953588	Vida de Rico	53	Camilo	Spain	MOVE_DOWN	115583.0	2021	8	13	3

5. SUPERVISED MACHINE LEARNING

STEP 1

I selected the features of interest and collected labeled training observations during the preprocessing of our dataset phase

STEP 2

the metrics chosen are:

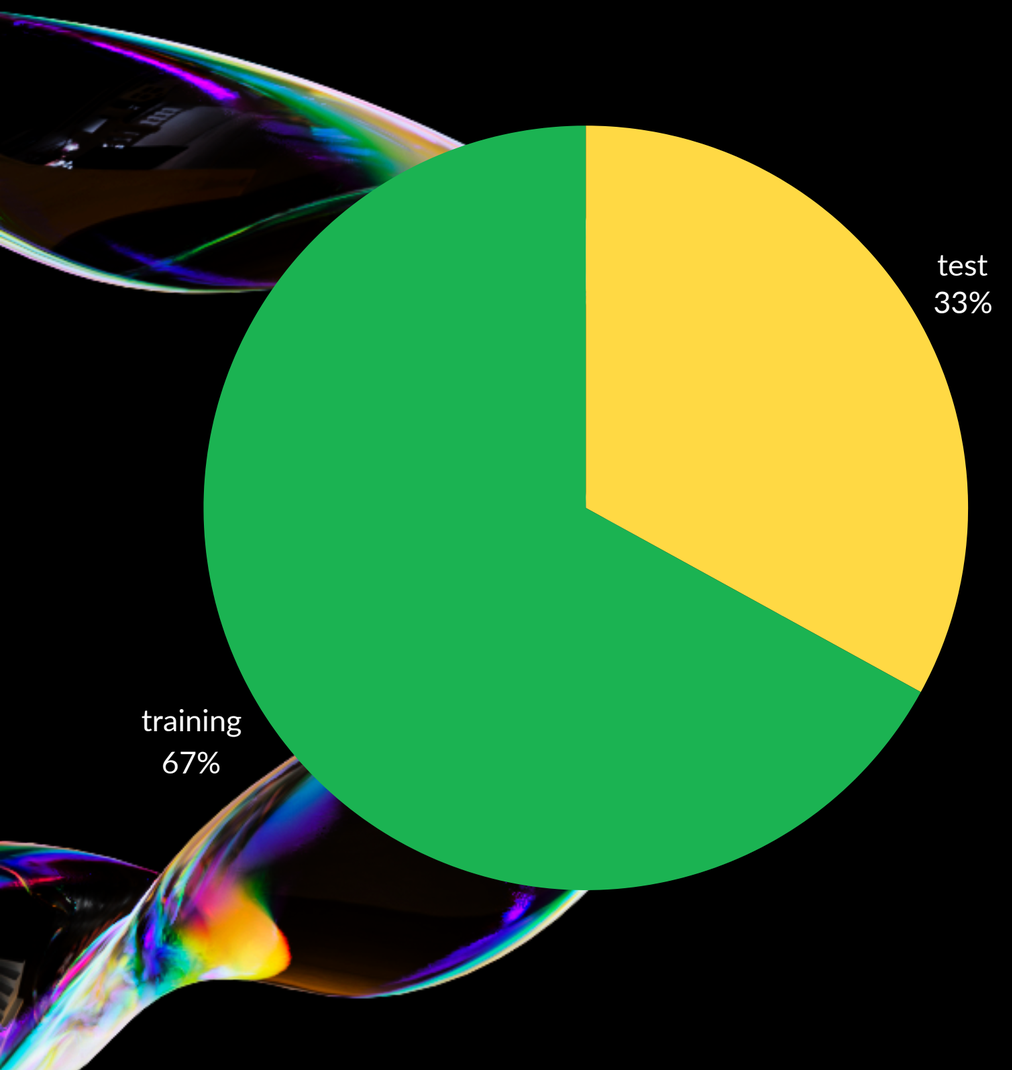
- accuracy score
- confusion matrix
- precision
- recall

STEP 3

the classification & optimization methods chosen are:

- decision tree
- random forest
- neural network

TRAIN & TEST SET



To split the data into training and testing sets we use the train_test_split method of scikit-learn library.

I split the X and Y dataset into :

TRAINING SET:

- dataset used to train the model
- the model learns to predict the output based on these input data

TESTING SET:

- dataset used to check the accuracy of the model
- the predictions made by the learning of the model using the unseen X feature are then compared with the true y test labels.

DECITION TREE

For classification problem: clustering

```
trend_clf=DecisionTreeClassifier()

trend_clf.fit(X,y)
```

```
DecisionTreeClassifier
DecisionTreeClassifier()
```

Dataset shape (1000000, 10)

X

Y

	rank	streams	year	month	day	season	artist_encoded	title_encoded	region_encoded	trend_encoded
9999818	75.0	3889.0	2017.0	8.0	11.0	3.0	8048.0	33301.0	33.0	3.0
9126206	191.0	7986.0	2020.0	4.0	4.0	2.0	1285.0	22094.0	9.0	4.0
14787372	2.0	43495.0	2020.0	8.0	7.0	3.0	5750.0	46035.0	60.0	3.0
5693904	143.0	25421.0	2019.0	4.0	25.0	2.0	7035.0	15310.0	29.0	1.0
22953588	53.0	115583.0	2021.0	8.0	13.0	3.0	3458.0	38799.0	57.0	3.0

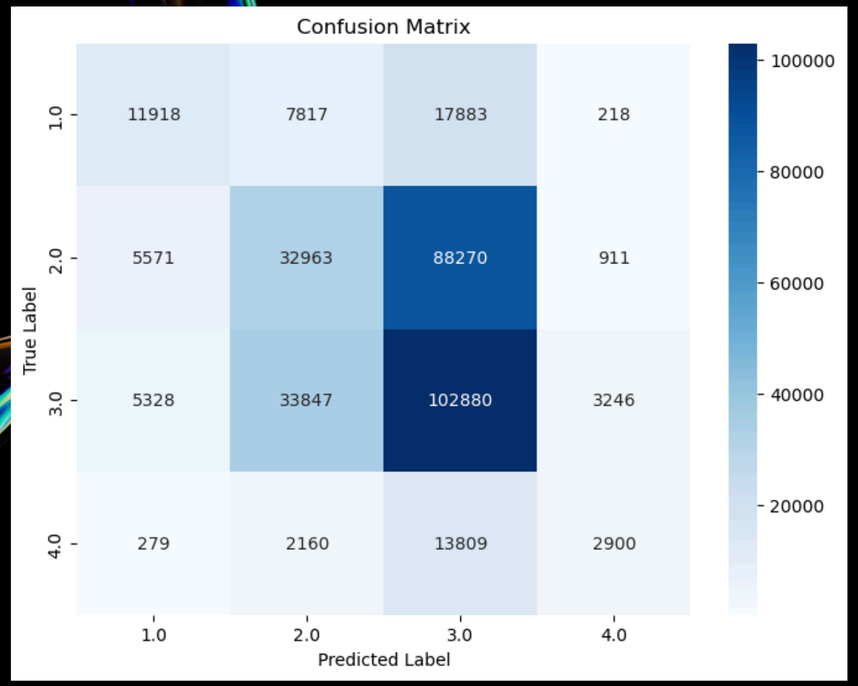
SOLVING OVERFITTING PROBLEM

Best parameters: {'max_depth': 15, 'max_leaf_nodes': 95000}
Best cross-validation accuracy: 0.457

Train Accuracy: 0.866
Test Accuracy: 0.418

X_train.shape: (670000, 9)
X_test.shape: (330000, 9)
y_train.shape: (670000,)
y_test.shape: (330000,)

Choosing the max leaves number and max depth of Decition Tree




```
for max_depth in range(15,30):
    # train and predict
    dt = tree.DecisionTreeClassifier(max_depth=max_depth)
    dt.fit(X_train,y_train)

    # compute Accuracy
    train_acc = accuracy_score(y_true = y_train, y_pred = dt.predict(X_train))
    test_acc  = accuracy_score(y_true = y_test,  y_pred = dt.predict(X_test))
    print ("Train Accuracy: {:.3f} - Test Accuracy: {:.3f}".format(train_acc,test_acc) )
```

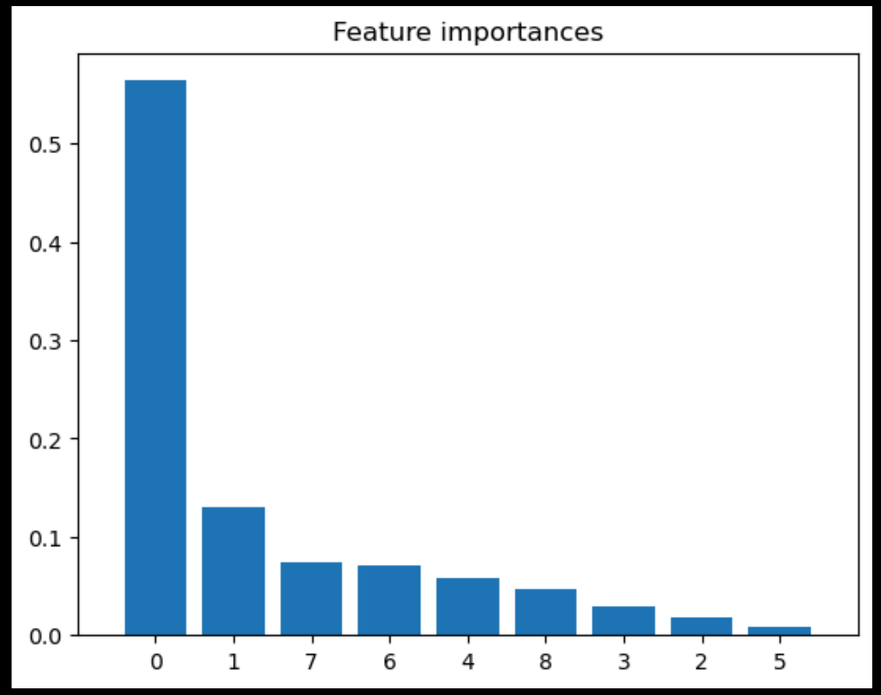
```
for max_leaves in range(90000,130000,1000):
    # train and predict
    dt = DecisionTreeClassifier(max_leaf_nodes=max_leaves)
    dt.fit(X_train,y_train)

    # compute Accuracy
    train_acc = accuracy_score(y_true = y_train, y_pred = dt.predict(X_train))
    test_acc  = accuracy_score(y_true = y_test,  y_pred = dt.predict(X_test))
    print ("Train Accuracy: {:.3f} - Test Accuracy: {:.3f}".format(train_acc,test_acc) )
```

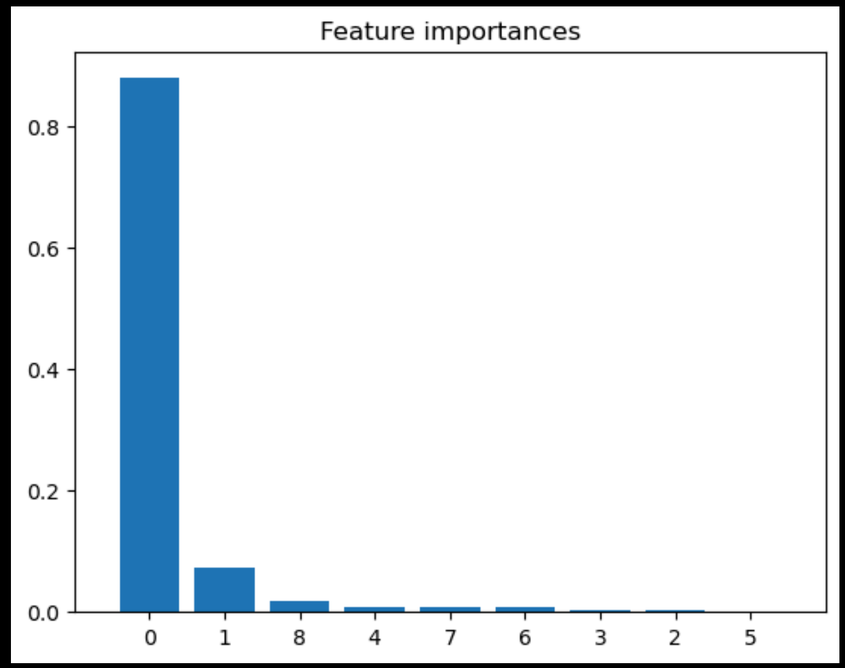
BEST PARAMETERS FROM GRIDSEARCHCV

Test Accuracy: 0.457
Classification Report:

	precision	recall	f1-score	support
1.0	0.52	0.32	0.39	37836
2.0	0.43	0.26	0.32	127715
3.0	0.46	0.71	0.56	145301
4.0	0.40	0.15	0.22	19148
accuracy			0.46	330000
macro avg	0.45	0.36	0.37	330000
weighted avg	0.45	0.46	0.43	330000

Confusion Matrix:
[[11935 7790 17894 217]
[5581 32955 88257 922]
[5340 33826 102876 3259]
[282 2154 13808 2904]]
Feature ranking:
1. Feature 0 (0.5637839053114143)
2. Feature 1 (0.1294227938333416)
3. Feature 7 (0.07390062696636893)
4. Feature 6 (0.07107904261586778)
5. Feature 4 (0.058667446268397636)
6. Feature 8 (0.04625482103691985)
7. Feature 3 (0.02890851257559356)
8. Feature 2 (0.018831101644602544)
9. Feature 5 (0.009151749747493606)





Best parameters: {'max_depth': 10, 'max_leaf_nodes': 90000}
Mean Squared Error: 0.489
R-squared Score: 0.174
Feature ranking:
1. Feature 0 (0.878179737199524)
2. Feature 1 (0.0733235525458665)
3. Feature 8 (0.017909522463957887)
4. Feature 4 (0.008347471893633836)
5. Feature 7 (0.007902804632181204)
6. Feature 6 (0.007667119352573152)
7. Feature 3 (0.004158602455225824)
8. Feature 2 (0.0021625982434038426)
9. Feature 5 (0.0003485912136336648)

RANDOM FOREST

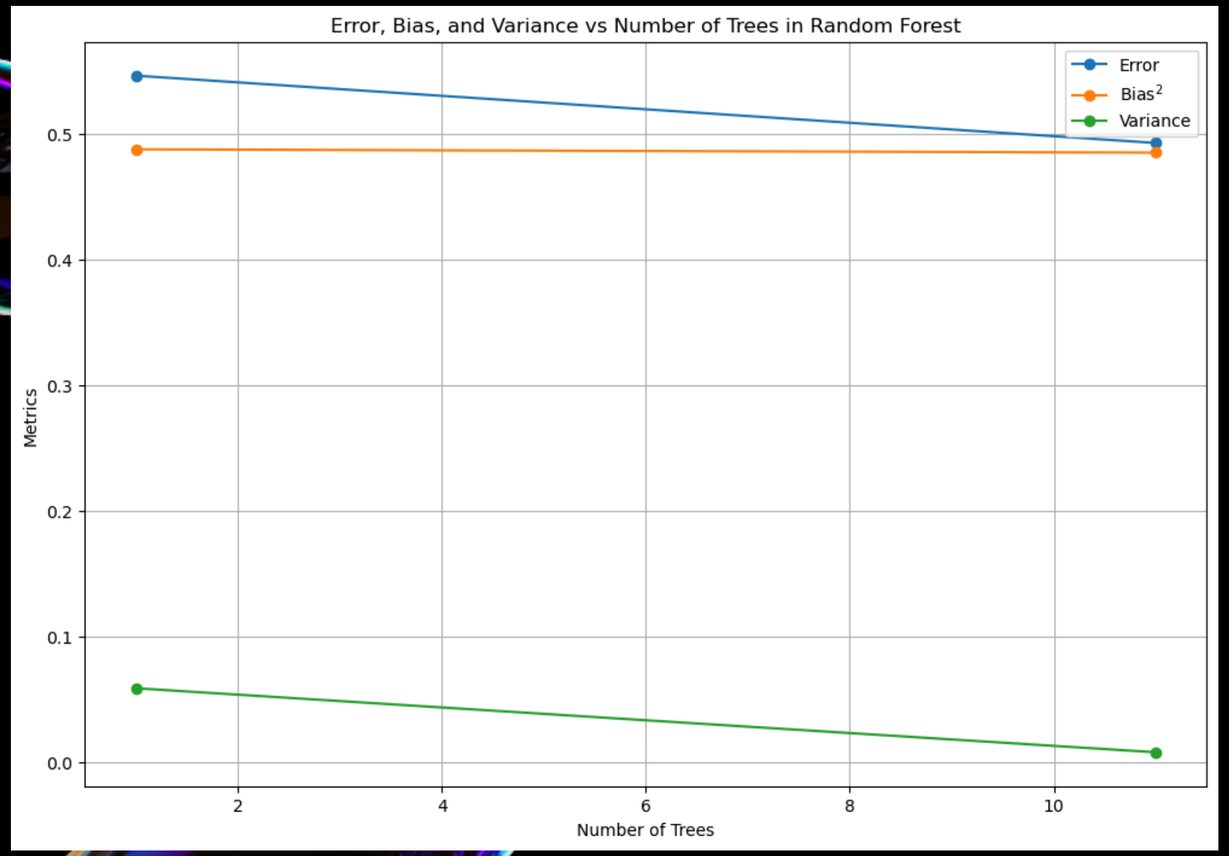
Drop unnecessary features

```
X_train1=X_train.drop(columns=['month','year', 'season'])
X_test1= X_test.drop(columns=['month','year', 'season'])
```

Initialize and train the Random Forest Classifier with the best parameters

```
Classification Report:
              precision    recall  f1-score   support

         1.0       0.55      0.34      0.42     37836
         2.0       0.46      0.14      0.22    127715
         3.0       0.46      0.84      0.60    145301
         4.0       0.47      0.09      0.14     19148

    accuracy                           0.47    330000
   macro avg       0.48      0.35      0.35    330000
weighted avg       0.47      0.47      0.41    330000

Confusion Matrix:
[[ 12885   4495  20382     74]
 [  5319  18421 103677    298]
 [  4938  16719 122137   1507]
 [   240    683  16582   1643]]
```

With the best parameters: {'max_depth': 15, 'max_leaf_nodes': 90000}



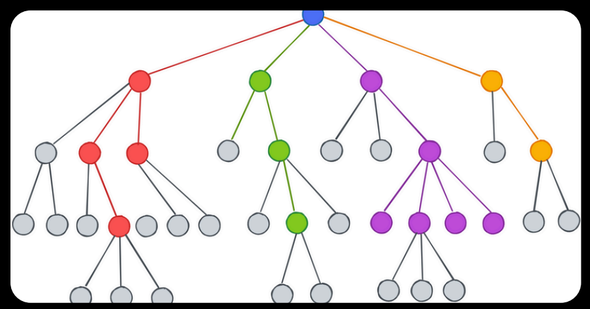
Final statistics for 11 trees

Error: 0.49259216957323404

Bias^2: 0.48489628527027107

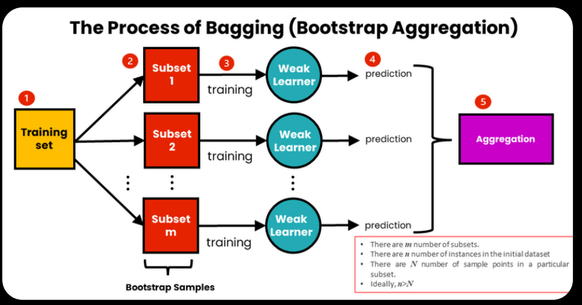
Variance: 0.007695884302962654

THE BEST ALGORITHM



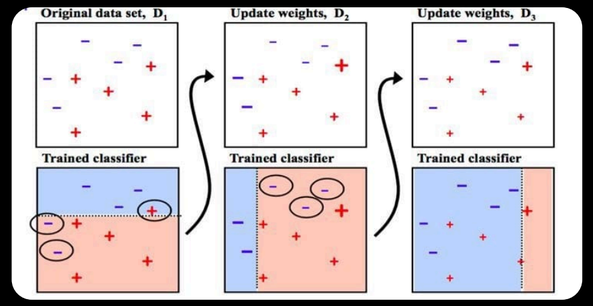
DECISION TREE

MSE: 0.4933



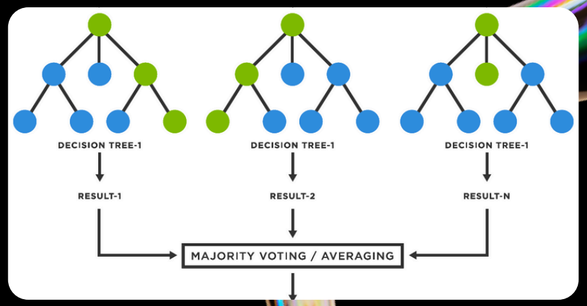
BAGGING

MSE: 0.4912



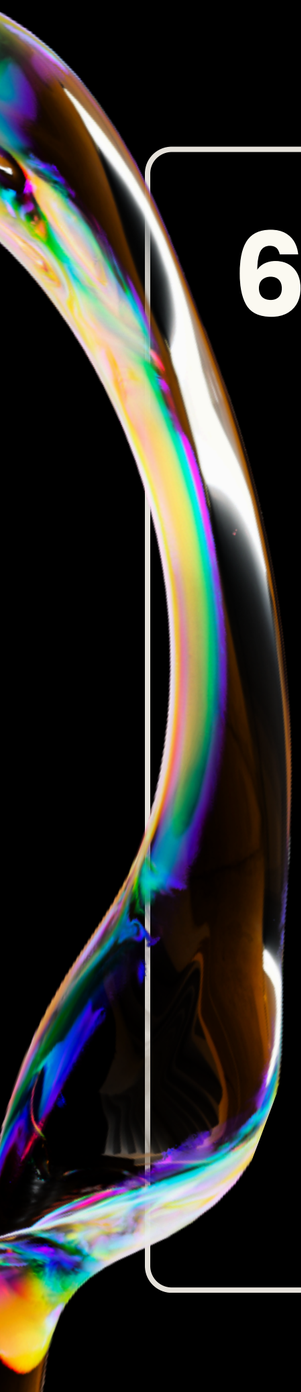
ADABOOST

MSE: 0.4976



RANDOMFOREST

MSE: 0.5038



6. NEURAL NETWORK

Labels range: 1.0 to 4.0
Number of unique classes in trend_encoded: 4

10313/10313 [==============================] - 2s
203us/step - loss: 1.0220 - accuracy: 0.4584

Test accuracy: 0.4583757519721985, Test loss:
1.0220179557800293

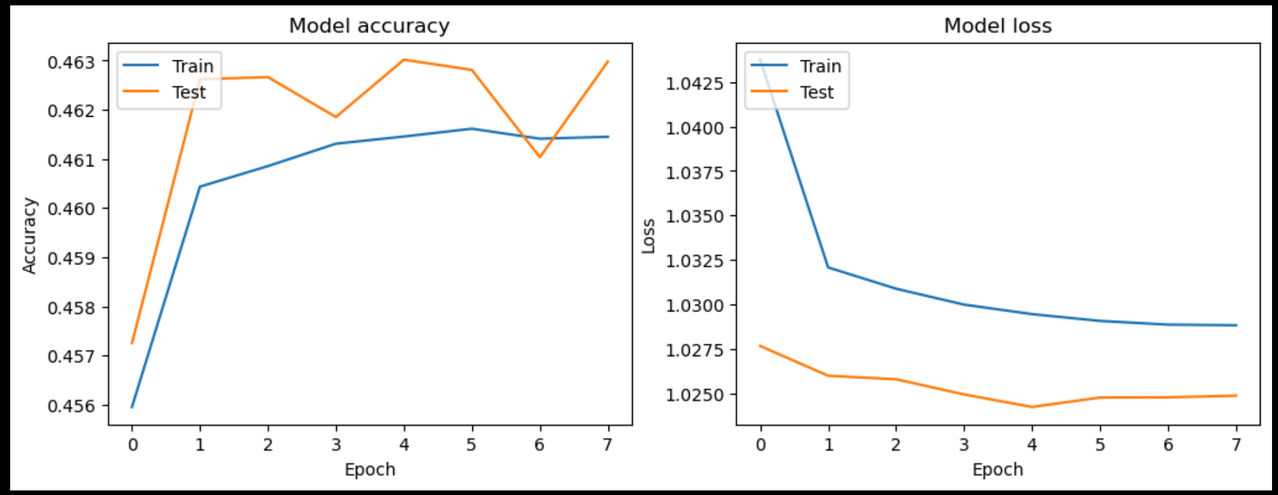
```
y = dataset['trend_encoded'] - 1
X = dataset.drop(columns=['trend_encoded'])


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)


X_train1 = X_train.drop(columns=['month', 'year', 'season'])
X_test1 = X_test.drop(columns=['month', 'year', 'season'])


scaler = StandardScaler()
X_train1 = scaler.fit_transform(X_train1)
X_test1 = scaler.transform(X_test1)


num_classes = len(y.unique())
print(f"Number of unique classes in trend_encoded: {num_classes}")


print(f"Labels range in y_train: {y_train.min()} to {y_train.max()}")
print(f"Labels range in y_test: {y_test.min()} to {y_test.max()}")


input_shape = X_train1.shape[1]
model = models.Sequential([
    layers.Dense(64, activation='relu', input_shape=(input_shape,)),
    layers.Dense(32, activation='relu'),
    layers.Dense(num_classes, activation='softmax')
])


model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

# Define training parameters
epochs = 10
batch_size = 32


history = model.fit(X_train1, y_train,
                    epochs=epochs,
                    batch_size=batch_size,
                    validation_data=(X_test1, y_test))


test_loss, test_acc = model.evaluate(X_test1, y_test)
print(f'Test accuracy: {test_acc}, Test loss: {test_loss}')
```

VALIDATION ACCURACY VALUES

```
input_shape = X_train1.shape[1]
model = models.Sequential([
    layers.Dense(128, activation='relu', input_shape=(input_shape,)),
    layers.Dropout(0.5),
    layers.Dense(64, activation='relu'),
    layers.Dropout(0.5),
    layers.Dense(num_classes, activation='softmax')
])


model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])


epochs = 20
batch_size = 32

# Train the model
history = model.fit(X_train1, y_train,
                    epochs=epochs,
                    batch_size=batch_size,
                    validation_data=(X_test1, y_test))

test_loss, test_acc = model.evaluate(X_test1, y_test)
print(f'Test accuracy: {test_acc}, Test loss: {test_loss}')
```

Test accuracy: 0.4629787802696228

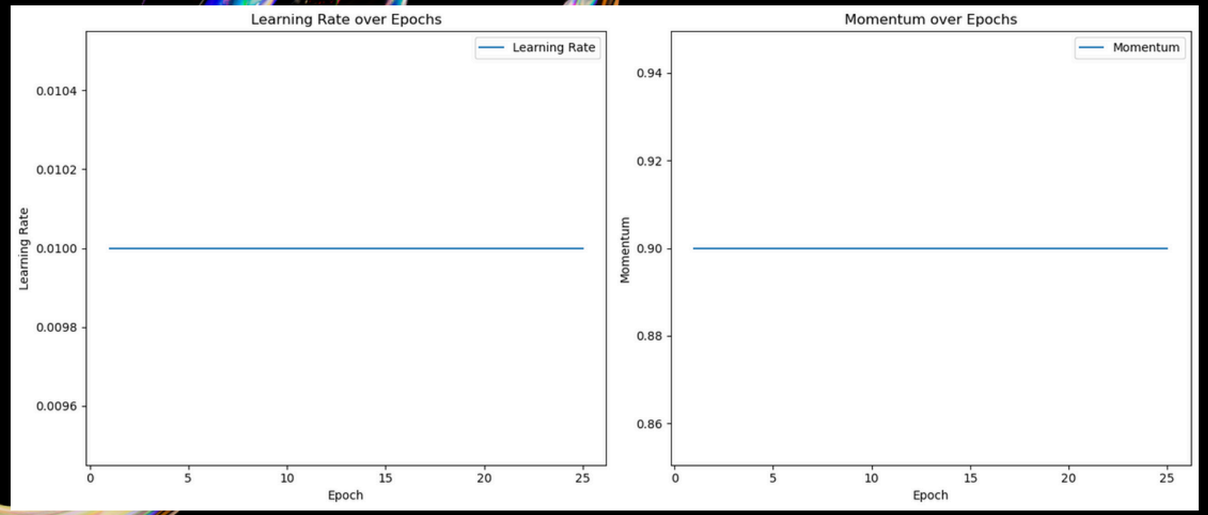
Test loss: 1.0248525142669678



	precision	recall	f1-score	support
0.0	0.52	0.34	0.42	37836
1.0	1.00	0.00	0.00	127715
2.0	0.46	0.96	0.62	145301
3.0	0.00	0.00	0.00	19148
accuracy			0.46	330000
macro avg	0.50	0.33	0.26	330000
weighted avg	0.65	0.46	0.32	330000

```
[[ 12989      0  24847      0]
 [  5979      1 121735      0]
 [  5508      0 139793      0]
 [   272      0  18876      0]]
```

KFOLD & LEARNINGRATEMOMENTUMLOGGER



Fold Test accuracy:
0.4648500084877014

Test loss: 1.022726058959961

Mean Test Accuracy from Cross-Validation: 0.4637820065021515

Test accuracy:
0.4652700126171112

Test loss: 1.021323323249817

```
scaler = StandardScaler()
X_train1 = scaler.fit_transform(X_train1)
X_test1 = scaler.transform(X_test1)


num_classes = len(y.unique())
print(f"Number of unique classes in trend_encoded: {num_classes}")


print(f"Labels range in y_train: {y_train.min()} to {y_train.max()}")
print(f"Labels range in y_test: {y_test.min()} to {y_test.max()}")


input_shape = X_train1.shape[1]
model = models.Sequential([
    layers.Dense(128, activation='relu', input_shape=(input_shape,)),
    layers.Dropout(0.3),
    layers.Dense(64, activation='relu'),
    layers.Dropout(0.3),
    layers.Dense(num_classes, activation='softmax')
])


model.compile(optimizer=optimizer,
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])


epochs = 30
batch_size = 64
early_stopping = EarlyStopping(monitor='val_loss', patience=3)
learning_rate_momentum_logger = LearningRateMomentumLogger(optimizer)


history = model.fit(X_train1, y_train,
                    epochs=epochs,
                    batch_size=batch_size,
                    validation_data=(X_test1, y_test),
                    callbacks=[early_stopping, learning_rate_momentum_logger])


test_loss, test_acc = model.evaluate(X_test1, y_test)
print(f'Test accuracy: {test_acc}, Test loss: {test_loss}')
```

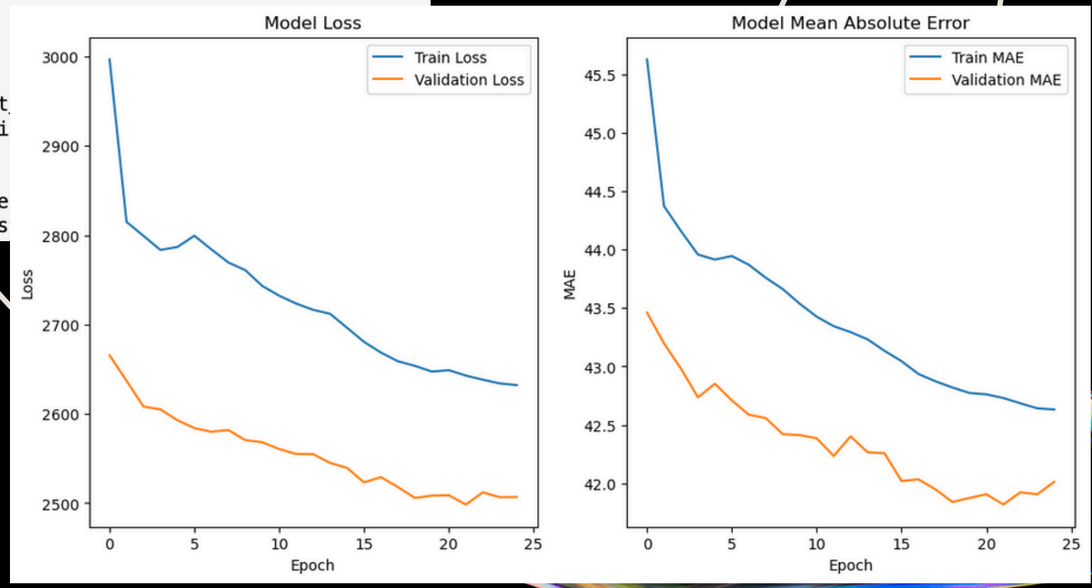
CHANGING Y: PREDICTION OF RANK

X_train_r1 shape: (670000, 6)
X_test_r1 shape: (330000, 6)
y_train_r shape: (670000,)
y_test_r shape: (330000,)

Test MAE: 42.01161575317383

Test loss: 2506.9052734375

```
input_shape = X_train_r1.shape[1]
model_r = models.Sequential([
    layers.Dense(128, activation='relu', input_shape=(input_shape,)),
    layers.Dropout(0.3),
    layers.Dense(64, activation='relu'),
    layers.Dropout(0.3),
    layers.Dense(1)
])


initial_learning_rate = 0.001
lr_schedule = tf.keras.optimizers.schedules.ExponentialDecay(
    initial_learning_rate,
    decay_steps=100000,
    decay_rate=0.96,
    staircase=True)

optimizer = Adam(learning_rate=lr_schedule, clipvalue=1.0)


model_r.compile(optimizer=optimizer,
                loss='mean_squared_error',
                metrics=['mean_absolute_error'])


epochs = 30
batch_size = 64
early_stopping = tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=3)


history_r = model_r.fit(X_train_r1, y_train_r,
                        epochs=epochs,
                        batch_size=batch_size,
                        validation_data=(X_test
                        callbacks=[early_stoppi


test_loss_r, test_mae_r = model_r.evaluate(X_te
print(f'Test MAE: {test_mae_r}, Test loss: {tes
```



7. CONCLUSION

Conclusion:

1. Data Quality and Quantity:
   - Limited and non-representative data likely hindered model accuracy.
   - Removing features like month, year, and season may have led to loss of crucial information.
2. Model Complexity and Hyperparameters:
   - The neural network architecture may need further tuning.
   - More optimization required for parameters such as learning rate and batch size.
3. Imbalanced Data:
   - Imbalanced target variables could have caused biased predictions.

Next Steps:

1. Enhance Data Quality: Collect more comprehensive data and improve preprocessing.
2. Revisit Feature Engineering: Include potentially valuable temporal data.
3. Model Optimization: Experiment with different architectures and fine-tune hyperparameters.
4. Explore Alternative Models: Compare performance with other machine learning models like Random Forest or Gradient Boosting.

Final Takeaway: While the neural network did not achieve the desired accuracy, the project provided valuable insights into the challenges of modeling chart success. Future efforts will focus on data enhancement, feature engineering, model tuning, and exploring alternative approaches to improve prediction accuracy and actionable insights.

Spotify

THANK YOU

for your time and attention