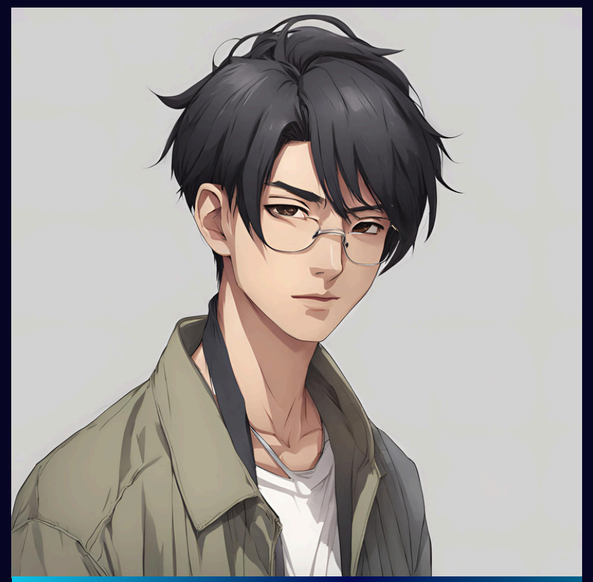
PROJECT_2

Regression analysis on Garment Workers productivity dataset

Tomorrow

MEET OUR TEAM

Jaiden Flint



Tuan Huynh

Kavita Gopal

Why is Productivity so important to the manufacturing industry?

Productivity reduces per unit manufacturing cost by effectively utilising all resources and reducing waste. Lower per-unit cost means an increase in profit level.

An economy is able to produce—and consume—increasingly more goods and services for the same amount of work.

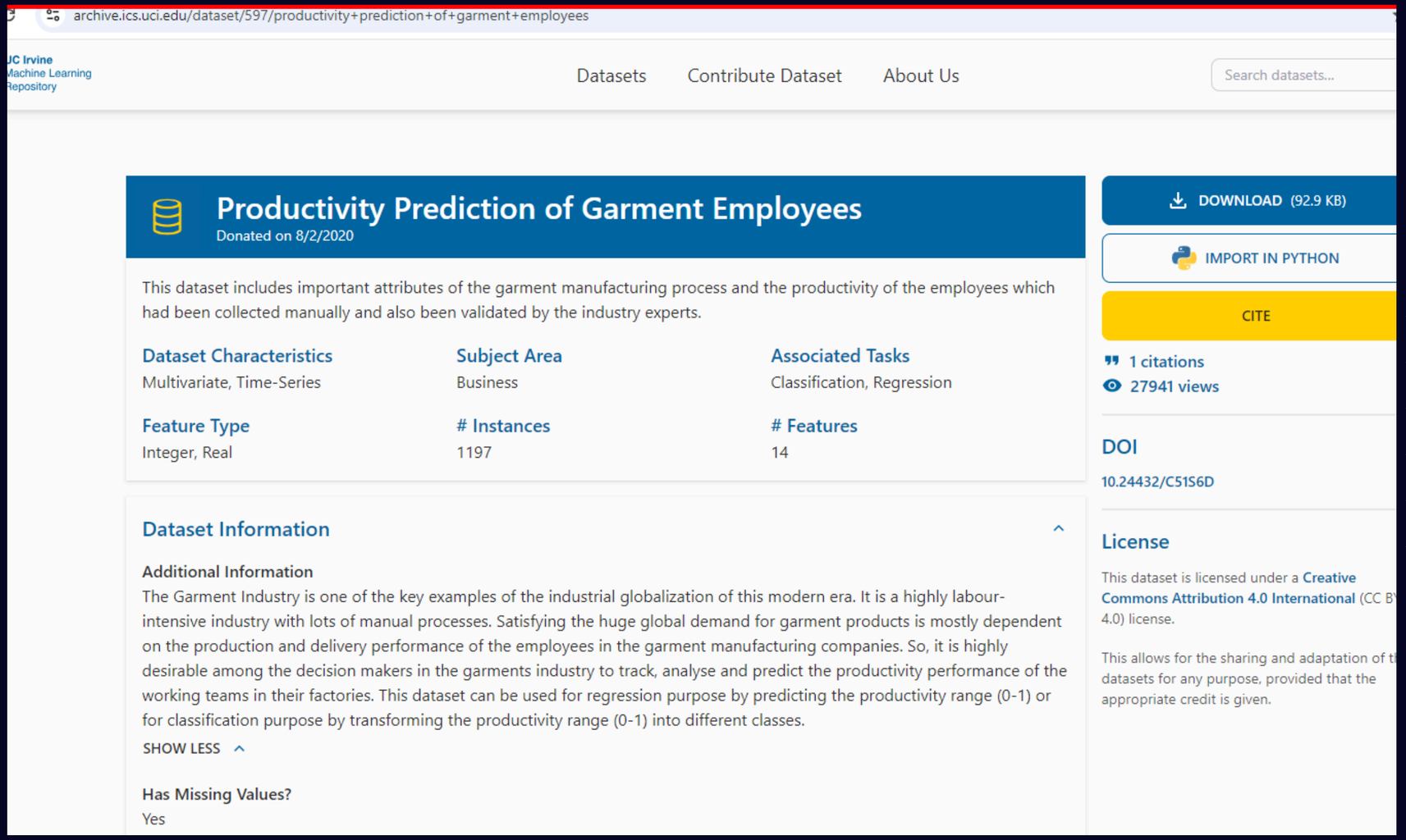
Productivity is important to workers, consumers, business leaders, and analysts involved in policy making and government statisticians.

Project Overview

This project aims to predict the productivity of garment employees based on features such as working hours, incentives, overtime, and more. The goal is to develop machine learning models that can accurately predict the continuous productivity score for garment workers, helping factories understand key drivers of productivity and optimize their operations.

OUR PROJECT

We used the UCI's ML repository to get our dataset.



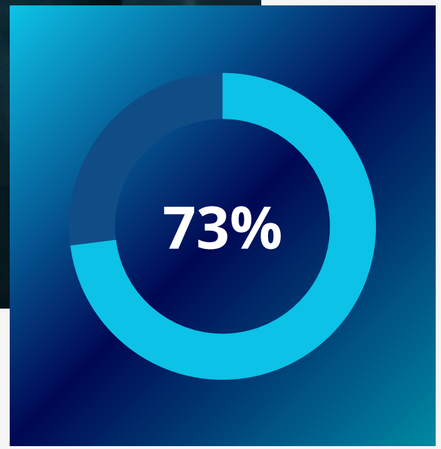
Dataset

- dataset Garment Worker Productivity
- consists of 1,197 rows and 15 features

1. date
2. quarter
3. department
4. day
5. team
6. targeted_productivity
7. smv (Standard Minute Value)
8. wip (Work In Progress)
9. over_time
10. incentive
11. idle_time
12. idle_men
13. no_of_style_change
14. no_of_workers
15. actual_productivity

- target variable: actual_productivity
- continuous primary variable
- we aim to predict using various machine learning models.

OUR PROJECT



Explortory Data Analysis (EDA)

```
# Display the updated DataFrame
updated_grpr_df
```

	date	quarter	department	day	team	targeted_productivity	smv	wip	over_time	incentive	idle_time	idle_men	no_of_style_change
0	1/1/2015	Quarter1	sewing	Thursday	8	0.80	26.16	1108.0	7080	98	0.0	0	0
1	1/1/2015	Quarter1	finishing	Thursday	1	0.75	3.94	NaN	960	0	0.0	0	0
2	1/1/2015	Quarter1	sewing	Thursday	11	0.80	11.41	968.0	3660	50	0.0	0	0
3	1/1/2015	Quarter1	sewing	Thursday	12	0.80	11.41	968.0	3660	50	0.0	0	0
4	1/1/2015	Quarter1	sewing	Thursday	6	0.80	25.90	1170.0	1920	50	0.0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
1192	3/11/2015	Quarter2	finishing	Wednesday	10	0.75	2.90	NaN	960	0	0.0	0	0
1193	3/11/2015	Quarter2	finishing	Wednesday	8	0.70	3.90	NaN	960	0	0.0	0	0
1194	3/11/2015	Quarter2	finishing	Wednesday	7	0.65	3.90	NaN	960	0	0.0	0	0
1195	3/11/2015	Quarter2	finishing	Wednesday	9	0.75	2.90	NaN	1800	0	0.0	0	0
1196	3/11/2015	Quarter2	finishing	Wednesday	6	0.70	2.90	NaN	720	0	0.0	0	0

1197 rows × 15 columns

```
# Print the list of columns to inspect the actual column names
print(cols)
```

```
['date', 'quarter', 'department', 'day', 'team', 'targeted_productivity', 'smv', 'wip', 'over_time', 'incentive', 'idle_time', 'idle_men', 'no
 of style change', 'no of workers', 'actual productivity', 'target achieved']
```

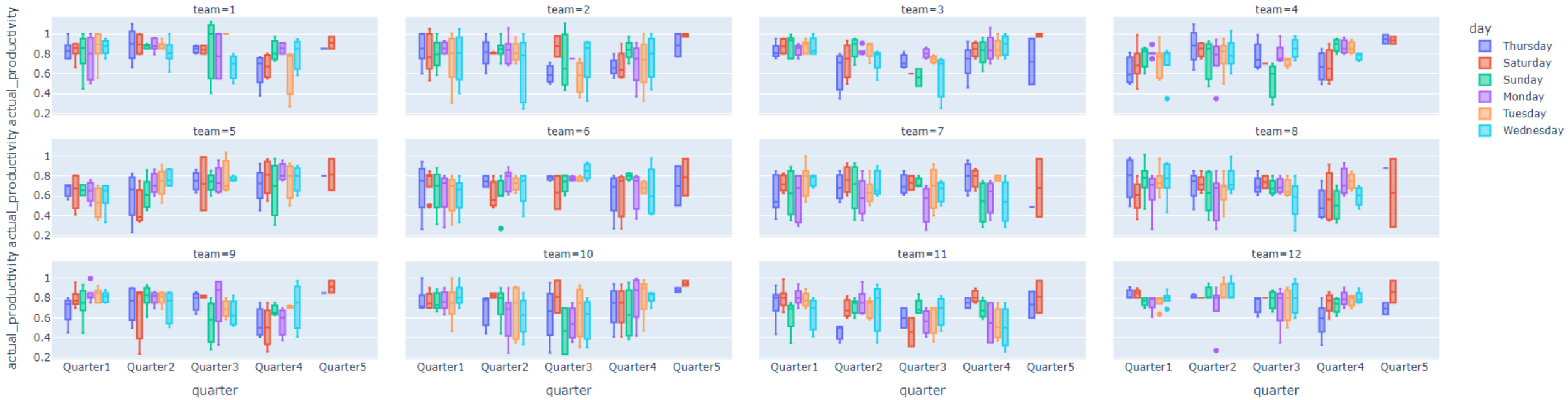
'smv',

']

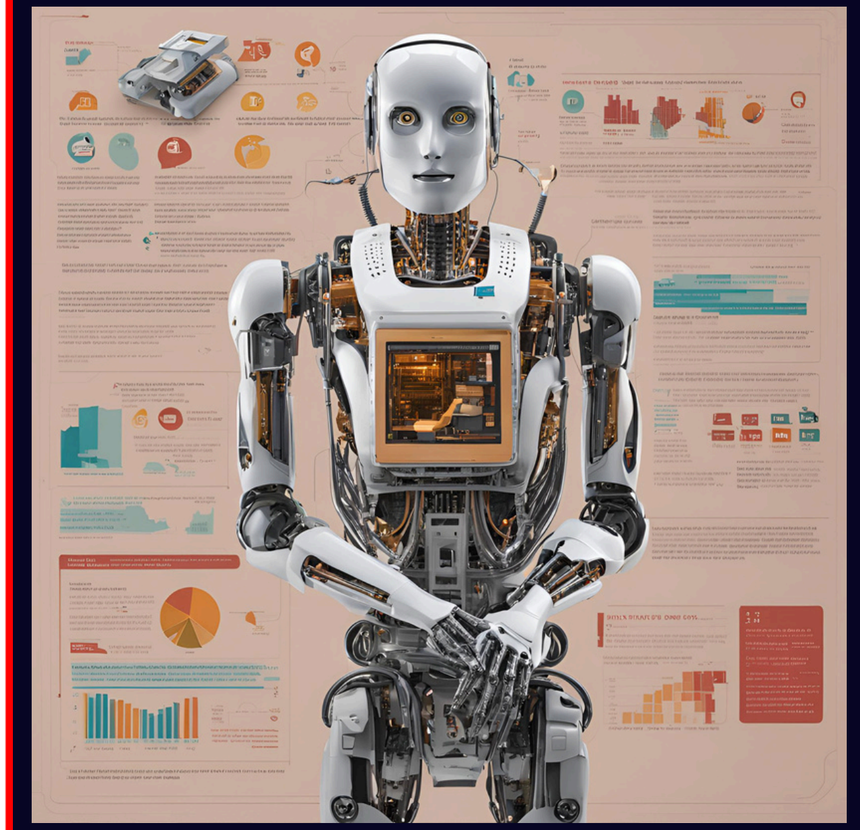
OUR PROJECT

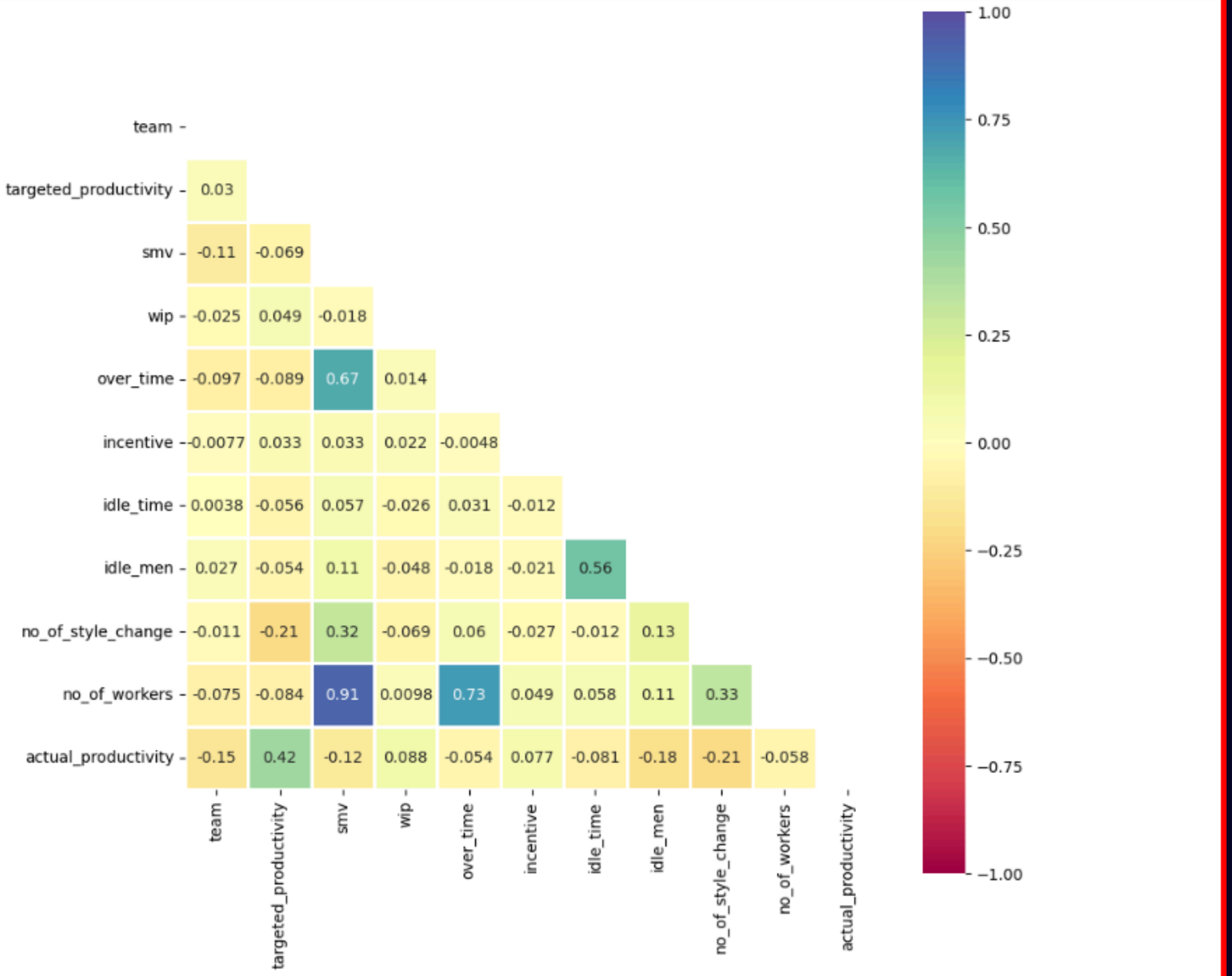
OUR VISION

```
# Create a list of teams in the desired order
team_order = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]

# Create the box plot with the specified order
fig = px.box(updated_grpr_df, x='quarter', y='actual_productivity', color='day',
             facet_col='team', facet_col_wrap=4, category_orders={"team": team_order})
fig.show()
```



OUR PROJECT





Correlation between features in the dataset: more incentives did not correlate to productivity as shown in the plot. No strong correlation when the value is closer to zero. Correlations vary from -1 to +1. Closer to +1 means strong positive correlation whereas closer to -1 means strong negative correlation. The variables with strong correlations are probably the features for model building.

OUR VISION

Data Cleaning and Processing

01

Handling Missing Values

- wip

02

Feature Selection

- actual_productivity, targeted_productivity removed to avoid data leakage
- date col removed: no relevant data for prediction

03

Categorical Feature Encoding

- One-Hot Encoding (scikit-learn): applied to categorical variables (quarter, department, and day).
- Transformation of categorical variables into numerical values suitable for ML.

04

Scaling Features

• Missing values in the features were filled using the mean, and all numeric features were scaled for better model performance.

OUR MISSION

Model Training

01

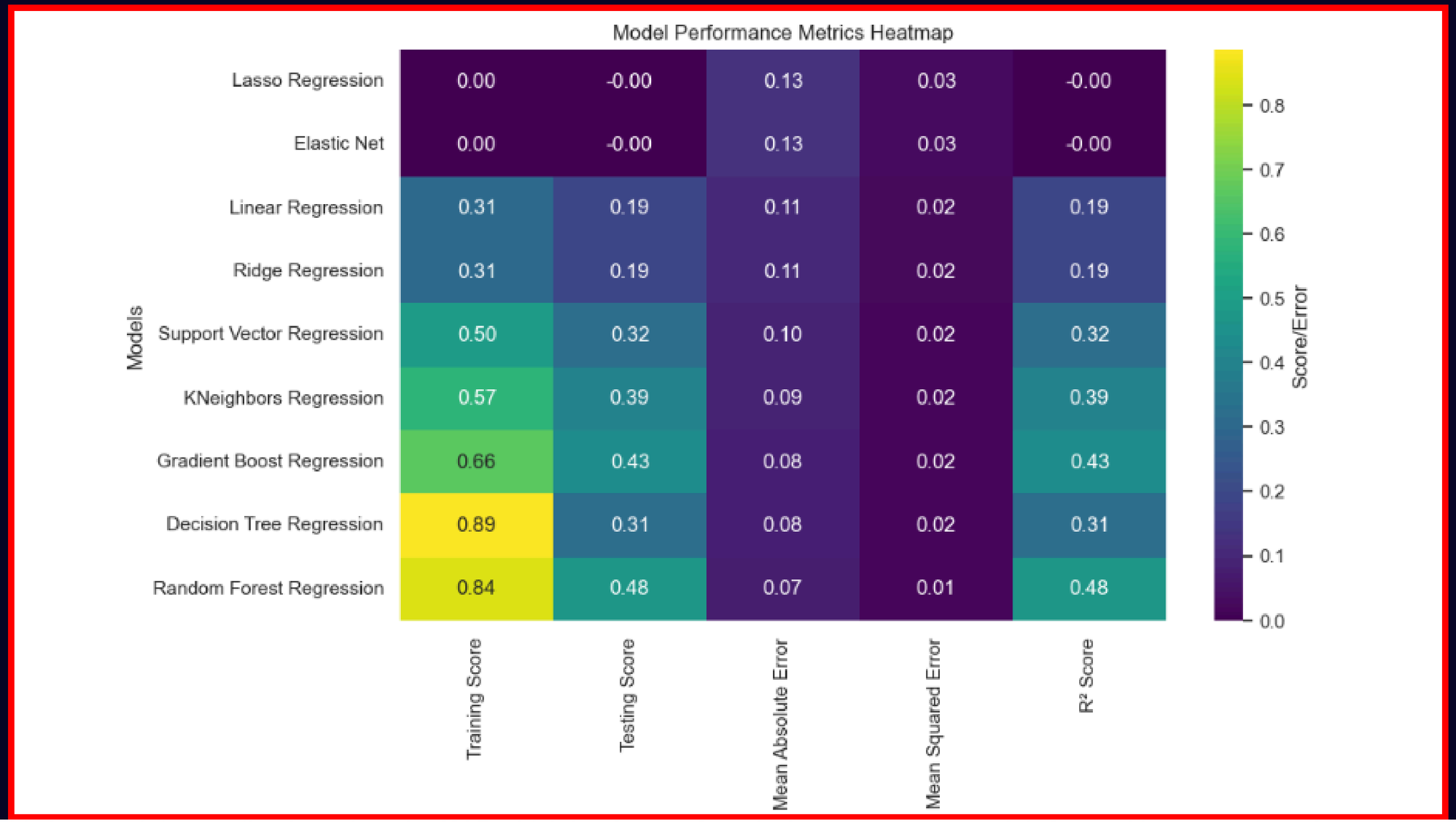
- Linear Regression
- Support Vector Regression
- KNeighbors Regression
- Gradient Boost Regression
- Random Forest Regression
- XgBoost Regression
- Decision Tree Regression
- Lasso Regression
- Elastic Net

02

Choice of Metric?

- MAE is often preferred when you want a straightforward interpretation of average error.
- MSE is useful when you want to penalize larger errors more heavily.
- R^2 is useful for understanding the explanatory power of the model, but it should not be the sole metric for evaluation.

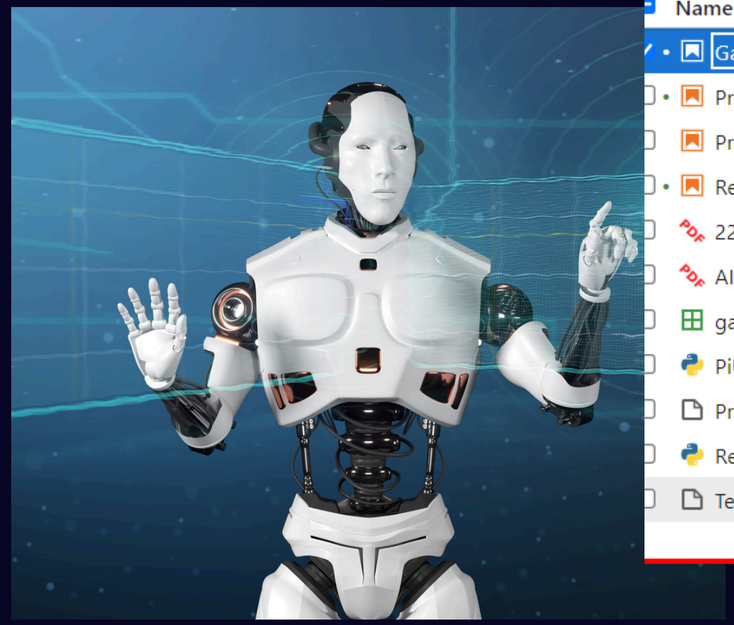
CRASH & BURN



The table is sorted descending based on the MAE which is often preferred when you want a straightforward interpretation of average error.

A lower MAE value indicates better model performance, as it means that the model's predictions are closer to the actual values on average. This is generally desirable in predictive modeling.

A higher MAE value suggests that the model's predictions are further away from the actual values, indicating poorer performance.



Name	Last Modified	File Size
GarPrPipelineUtils.ipynb	1 minute ago	5.7 MB
Project-2 Garment Productivity Data_test.ipynb.ipynb	10 minutes ago	2.4 MB
Project-2 Garment Productivity Data.ipynb.ipynb	yesterday	5.9 MB
Regression_Model_Project2.ipynb	2 hours ago	1.6 MB
2210-Article Text-5760-2-10-20210526.pdf	5 days ago	1.1 MB
AI-Overview2.drawio.pdf	2 days ago	86.1 KB
garments_worker_productivity.csv	7 days ago	92.7 KB
PiUtils.py	21 hours ago	4.1 KB
Project2.docx	4 days ago	41.4 KB
Regression_Model_Pipeline.py	2 hours ago	5.5 KB
Template Readme for Project 2.docx	yesterday	13.4 KB

We had to start-over and rethink the startegy to clean the data for more menaningful model training!!!

	Original					Fitted				
	Training Score	Testing Score	MAE	MSE	R² Score	Training Score	Testing Score	MAE	MSE	R² Score
Lasso Regression	0.000000	-0.000685	0.128256	0.028178	-0.000685	0.254333	0.170077	0.109296	0.023370	0.170077
Elastic Net	0.000000	-0.000685	0.128256	0.028178	-0.000685	0.303058	0.193400	0.108250	0.022713	0.193400
Linear Regression	0.306007	0.188046	0.109552	0.022864	0.188046	0.306007	0.188046	0.109552	0.022864	0.188046
Ridge Regression	0.305997	0.188579	0.109464	0.022849	0.188579	0.305275	0.191695	0.108875	0.022761	0.191695
Support Vector Regression	0.496357	0.317175	0.095912	0.019228	0.317175	0.496357	0.317175	0.095912	0.019228	0.317175
KNeighbors Regression	0.574632	0.388269	0.086326	0.017226	0.388269	0.514701	0.323550	0.089639	0.019048	0.323550
Gradient Boost Regression	0.663522	0.434893	0.081950	0.015913	0.434893	0.737412	0.401290	0.082453	0.016859	0.401290
Decision Tree Regression	0.885893	0.314978	0.079652	0.019290	0.314978	0.748720	0.345696	0.082957	0.018425	0.345696
Random Forest Regression	0.840723	0.476603	0.074872	0.014738	0.476603	0.736627	0.454106	0.078817	0.015372	0.454106

- retrun to the drawing board
- do something different in analysizing the data
- recreate the pipeline!!!
- try hyperparameter testing???

Metrics of the dataset.

First run through of the Regression Models.

REMEMBER NAN VALUES WERE FILLED WITH MEAN VALUES

```
X_train shape: (897, 10)
y_train shape: (897, 1)
X_test shape: (300, 10)
y_test shape: (300, 1)
```

	Training Score	Testing Score	Mean Absolute Error	Mean Squared Error	R^2 Score
Lasso Regression	0.000000	-0.000685	0.128256	0.028178	-0.000685
Elastic Net	0.000000	-0.000685	0.128256	0.028178	-0.000685
Linear Regression	0.306007	0.188046	0.109552	0.022864	0.188046
Ridge Regression	0.305997	0.188579	0.109464	0.022849	0.188579
Support Vector Regression	0.496357	0.317175	0.095912	0.019228	0.317175
KNeighbors Regression	0.574632	0.388269	0.086326	0.017226	0.388269
Gradient Boost Regression	0.663522	0.434893	0.081950	0.015913	0.434893
Decision Tree Regression	0.885893	0.314978	0.079652	0.019290	0.314978
Random Forest Regression	0.840723	0.476603	0.074872	0.014738	0.476603

Metrics of the dataset.

After Hyper Parameter Tuning.

```
X_train shape: (897, 10)
y_train shape: (897, 1)
X_test shape: (300, 10)
y_test shape: (300, 1)
```

	Training Score	Testing Score	Mean Absolute Error	Mean Squared Error	R^2 Score
Linear Regression	0.306007	0.188046	0.109552	0.022864	0.188046
Lasso Regression	0.254333	0.170077	0.109296	0.023370	0.170077
Ridge Regression	0.305275	0.191695	0.108875	0.022761	0.191695
Elastic Net	0.303058	0.193400	0.108250	0.022713	0.193400
Support Vector Regression	0.496357	0.317175	0.095912	0.019228	0.317175
KNeighbors Regression	0.514701	0.323550	0.089639	0.019048	0.323550
Decision Tree Regression	0.748720	0.345696	0.082957	0.018425	0.345696
Gradient Boost Regression	0.737412	0.401290	0.082453	0.016859	0.401290
Random Forest Regression	0.736627	0.454106	0.078817	0.015372	0.454106

PIPELINE.PY

```python
# ------------------------------------------------------------------------------
# Hyper Parameter Tuning Using Grid Search
# ------------------------------------------------------------------------------

def tune_models(pipelines, param_grids, X_train, y_train):
    results = {}
    for name, pipeline in pipelines.items():
        print(f"Tuning {name}...")
        grid_search = GridSearchCV(estimator=pipeline, param_grid=param_grids[name], cv=5, scoring='neg_mean_squared_error')
        grid_search.fit(X_train, y_train)
        results[name] = {
            'best_params': grid_search.best_params_,
            'best_score': grid_search.best_score_
        }
    return results
```

Used Grid Search CV for Hyper Parameter Tuning of the data.
Then re-ran the models with the best parameters.

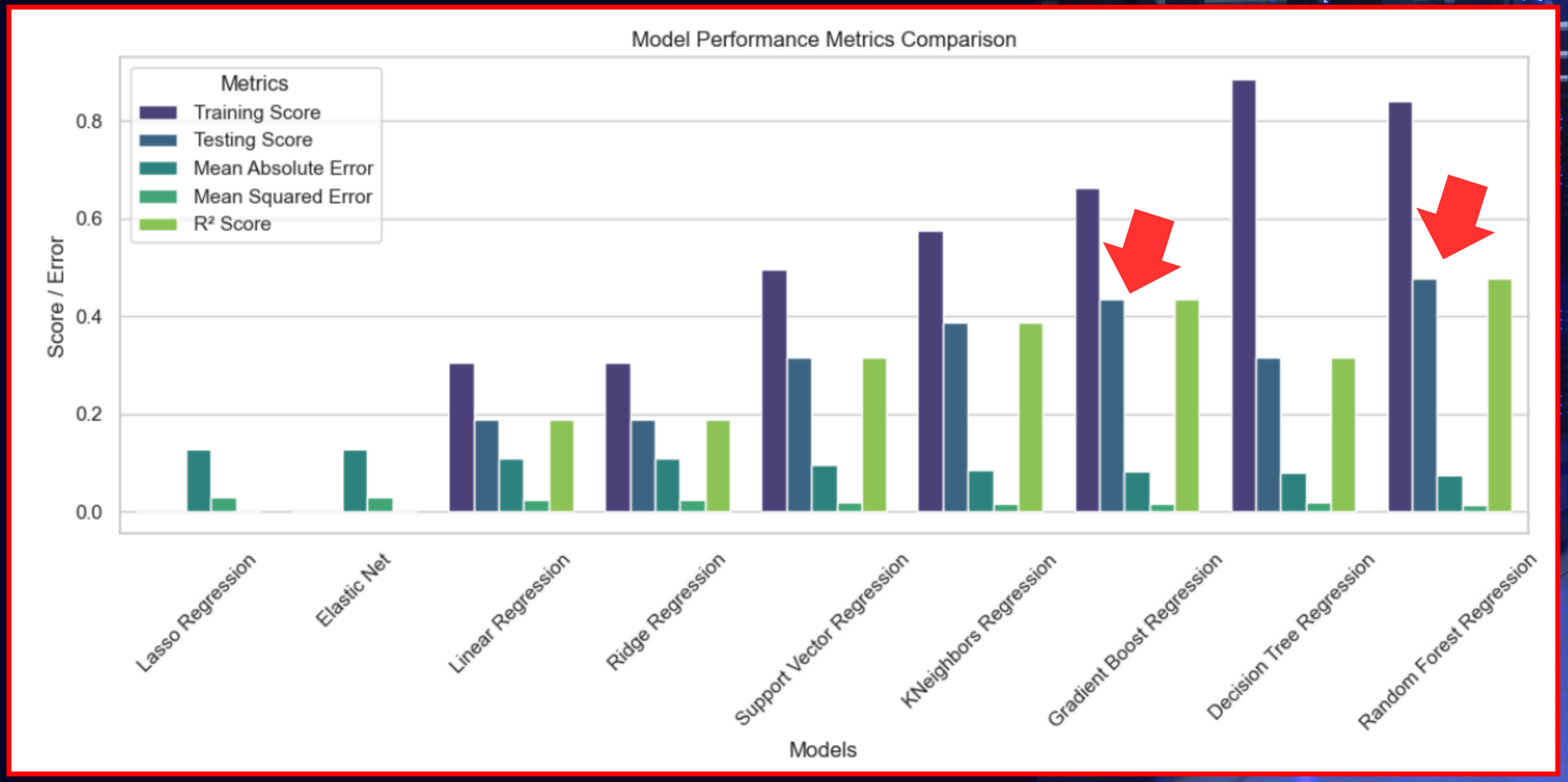
MAIN SCRIPT

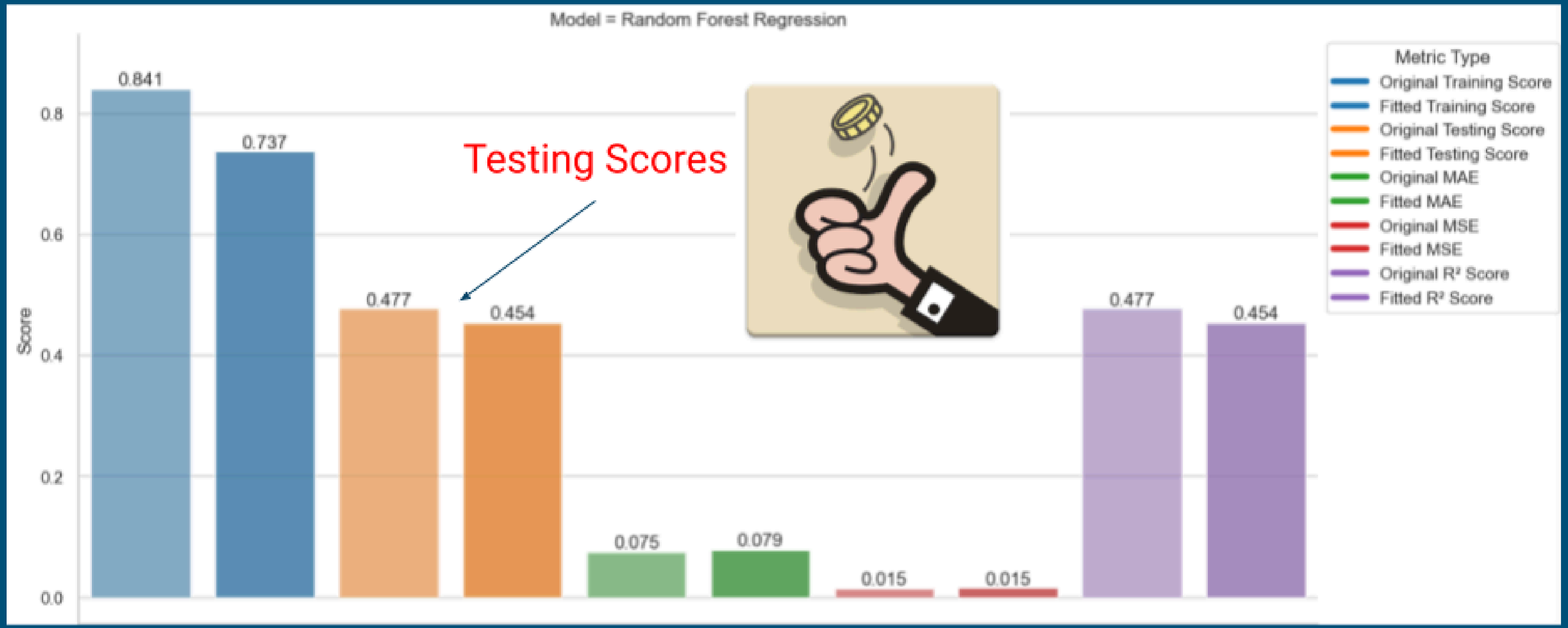
```python
fitted_models = {}

for model_name, result in tuning_results.items():
    best_params = result['best_params']
    fitted_model = create_and_fit_model(model_name, best_params, X_train, y_train, pipelines)
    fitted_models[model_name] = fitted_model  # Store the fitted model
```

```python
fitted_metrics_df = evaluate_models(fitted_models, X_train, y_train, X_test, y_test)
```

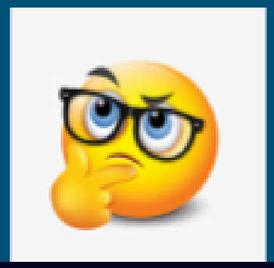
ABOUT FUTURE TECHNOLOGY

In this dataset the Tree models: Random Forest and Gradient Boost Regression Models had the best results for model training and testing for the first run through.





Scores don't look promising for trying to predict anything after training and testing the data.



EVEN AFTER HYPER PARAMETER TUNING!!!

Time to rethink what else can we do or fix the dataset?

Feature Selection: Evaluate the features used in the model.

Data Quality: Check for missing values, outliers, or errors in the dataset.

Model Complexity: Consider whether the model is too simple (underfitting) or too complex (overfitting).

Bias-Variance Trade-off: Assess the bias and variance of your model. If your model has high bias, you may need a more complex model. If it has high variance, you may need to simplify the model or use techniques like regularization.

Cross-Validation: Implement cross-validation to ensure that your model's performance is consistent across different subsets of the data. This helps to validate that your model is not just performing well on a specific training set.

Evaluation Metrics: Use appropriate evaluation metrics that align with your goals. For example, if your data is imbalanced, accuracy might not be the best metric; consider using metrics like R-squared, adjusted R-squared, or mean squared error (MSE).

Hyperparameter Tuning: Experiment with different hyperparameters to find the optimal settings for your model.

Metrics of the dataset.

Dataset with NaN values filled with zeros.

NaN = 0

```
X_train shape: (897, 10)
y_train shape: (897, 1)
X_test shape: (300, 10)
y_test shape: (300, 1)
```

	Training Score	Testing Score	Mean Absolute Error	Mean Squared Error	R^2 Score
Lasso Regression	0.000000	-0.000685	0.128256	0.028178	-0.000685
Elastic Net	0.000000	-0.000685	0.128256	0.028178	-0.000685
Linear Regression	0.305653	0.187449	0.109557	0.022881	0.187449
Ridge Regression	0.305644	0.187981	0.109469	0.022866	0.187981
Support Vector Regression	0.491446	0.315339	0.096369	0.019279	0.315339
Decision Tree Regression	0.885893	0.168972	0.087337	0.023401	0.168972
KNeighbors Regression	0.574089	0.389248	0.086383	0.017198	0.389248
Gradient Boost Regression	0.667752	0.426369	0.082126	0.016153	0.426369
Random Forest Regression	0.838553	0.453764	0.076116	0.015382	0.453764

```
The dataset is modified with the NaN values being filled with 0
```

Metrics of the dataset.

Dataset with NaN values being dropped.

NaN = Drop

```
X_train shape: (518, 10)
y_train shape: (518, 1)
X_test shape: (173, 10)
y_test shape: (173, 1)
```

	Training Score	Testing Score	Mean Absolute Error	Mean Squared Error	R^2 Score
Elastic Net	0.000000	-0.000209	0.113491	0.023086	-0.000209
Lasso Regression	0.224467	0.233420	0.098067	0.017694	0.233420
Support Vector Regression	0.826962	0.704936	0.060286	0.006811	0.704936
Decision Tree Regression	1.000000	0.661861	0.043695	0.007805	0.661861
Linear Regression	0.796404	0.828054	0.042139	0.003969	0.828054
Ridge Regression	0.796402	0.827945	0.042128	0.003971	0.827945
KNeighbors Regression	0.852334	0.838551	0.036810	0.003726	0.838551
Gradient Boost Regression	0.946769	0.851488	0.034835	0.003428	0.851488
Random Forest Regression	0.972617	0.830805	0.033738	0.003905	0.830805

```
The dataset is modified with the NaN values being dropped
```

THANK YOU

FOR YOUR PARTICIPATION!