1. Goal

The goal of the project is to analyze the errors of the linear regression model, identify possible reasons for the decrease in quality and test improved architectures (polynomial regression, hybrid approaches) to increase the accuracy of predictions.

The project had two main steps:

1. **Model diagnostics**
   - Check how well the simple linear model fits the data
   - Look for patterns in the residuals that might indicate missing relationship or non-linearity.

2. **Improving performance**
   - Try mode flexible models (polynomial features, tree-based models).
   - Compare their performance using metrics and error analysis.

What I've done:

+ Proposed hypothesis that the model's errors were caused by non-linear relationships in data.
+ Analysed this by analyzing the residuals.
+ Experimented a few models
+ Compared results and did conclusions.

2. Data. Info

Data is from Kaggle-Playground where the prediction goal was predict the number of calories a person has burnt during a workout based on some biological measures.
https://www.kaggle.com/c/playground-series-s5e5/data

info :

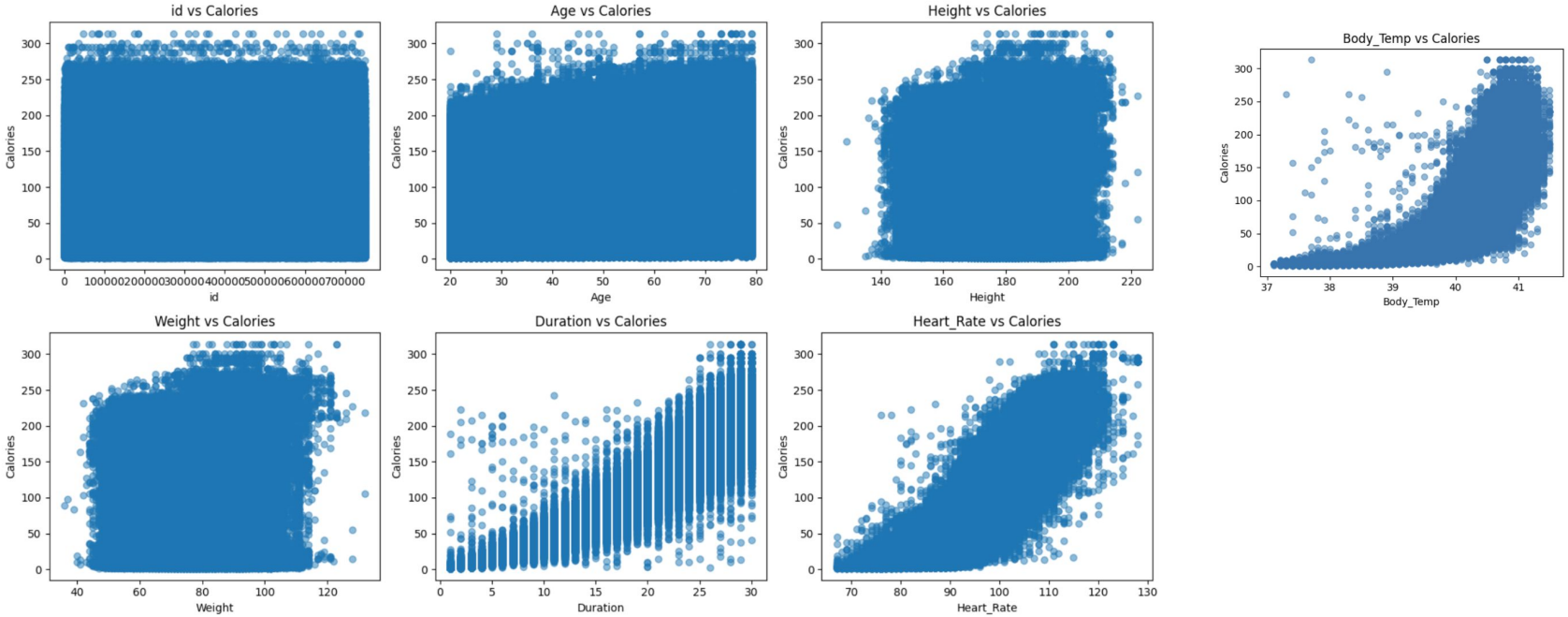
Size: 70,000 observations, 8 features
Data types: 7 numeric, 1 categorical
Target: 'Calories'
Brief statistics:

- Age from 20 to 79
- Height from 126 to 222
- Weight from 36 to 132
- Sex: female 375721
  male 374279

the categories are distributed almost evenly.

	id	Sex	Age	Height	Weight	Duration	Heart_Rate	Body_Temp	Calories
0	0	male	36	189.0	82.0	26.0	101.0	41.0	150.0
1	1	female	64	163.0	60.0	8.0	85.0	39.7	34.0
2	2	female	51	161.0	64.0	7.0	84.0	39.8	29.0
3	3	male	20	192.0	90.0	25.0	105.0	40.7	140.0
4	4	female	38	166.0	61.0	25.0	102.0	40.6	146.0

2. Data. Features vs Target Correlation



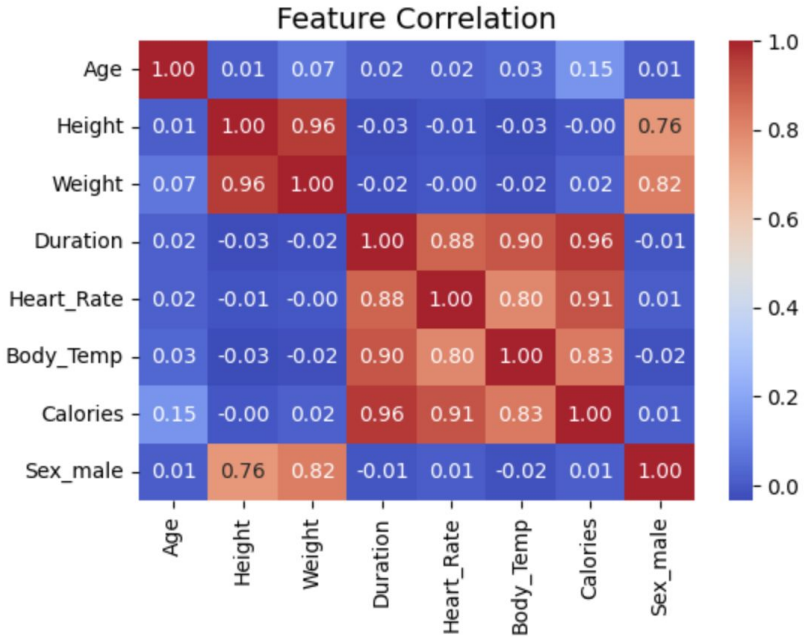
Let's analyze the information received:
- Age vs Calories , Height vs Calories, Weight vs Calories - no obvious relationship.
- Duration vs Calories, Heart_Rate vs Calories, Body_tem vs Calories - the high correlation that we noticed earlier was confirmed.

Else I see some non-linear pattern in those high-correlated data. To analyse it I will try to fit it by some polynomial function or something else, but for the simple baseline let's check linear function to high-correlated features.

2. Data. Multicollinearity features

Duration, Heart_Rate = 0.88
Duration, Body_Temp = 0.90
Duration, Calories = 0.96
Heart_Rate, Body_Temp = 0.80
Heart_Rate, Calories = 0.91
Body_Temp, Calories = 0.83

Height vs Weight = 0.96

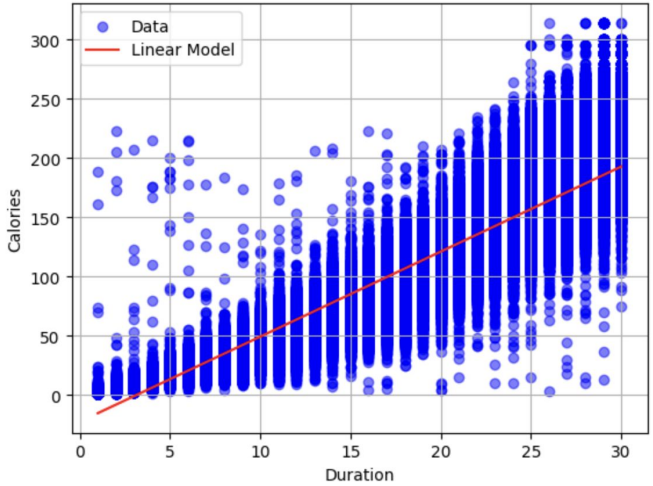


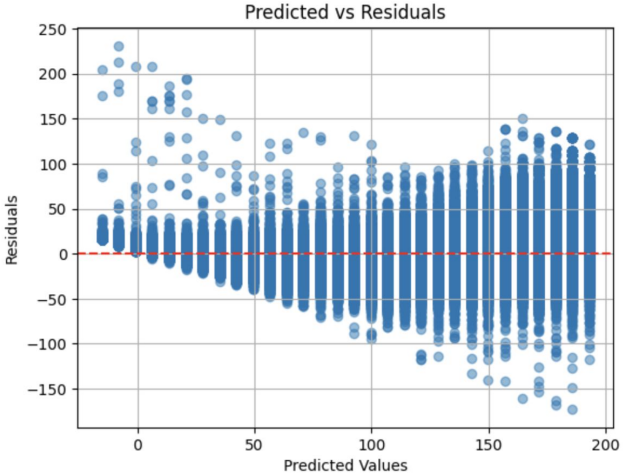
The analysis showed a high correlation between the features (0.8–0.96), which indicates multicollinearity. This is acceptable for forecasting, but for analyzing the influence of individual factors, the results should be interpreted with caution

So it gives us an idea to predict 'Calories' we can try to train a model on one feature (for example, Duration) and compare the quality (R^2, MAE, RMSE) with a model on three features

3. Prediction. Modeling

Hypothesis: The relationship between features and the target can be adequately modeled using a simple linear regression with only one the most correlated feature. Less is better.





Heteroscedasticity
There is an uneven spread of errors

Model with 1 feature:

```
X = data[['Duration']]
y = data['Calories']

model = LinearRegression() # y = w0 + w1 * 'Duration'
```

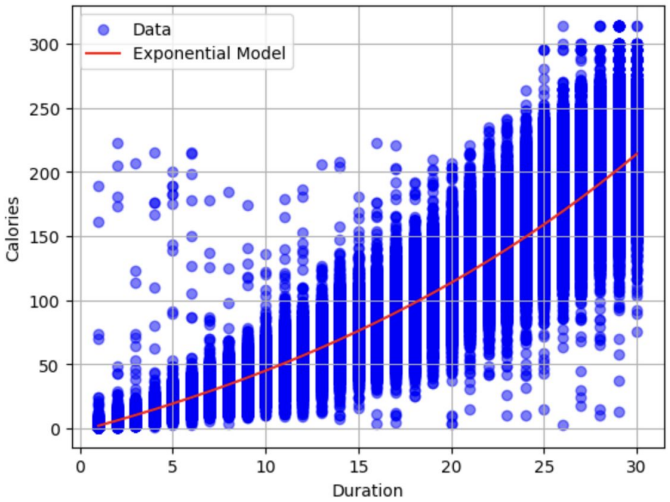
MSE: 302.57
RMSE: 17.39

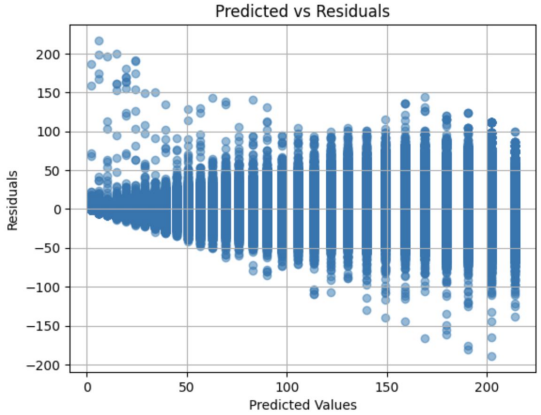
What we can do to reduce systematic model's errors?

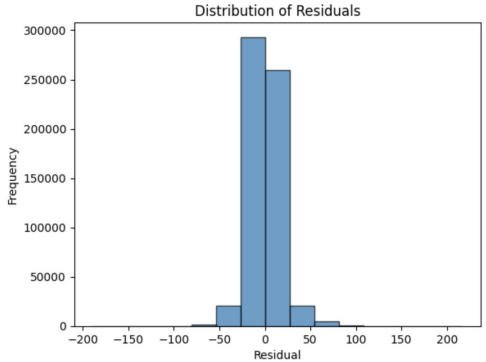
3. Prediction. Exponential Model

The idea behind improving the simple linear model on a single feature is to try to capture the nonlinearity using an exponential model.

The exponential model was build : y = 100.77 * exp(0.0382 * x) + -102.62



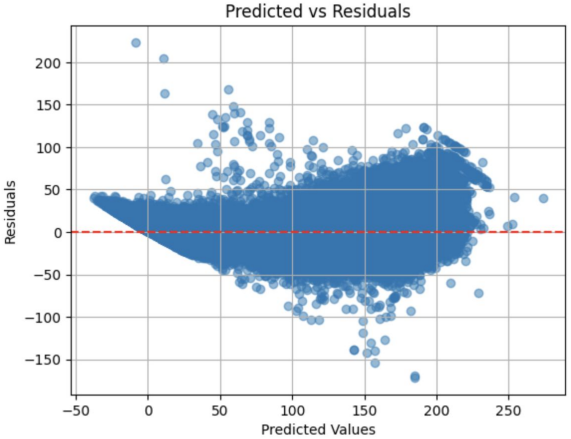




Model with 3 features:
y = c + a * exp(b * 'Duration')

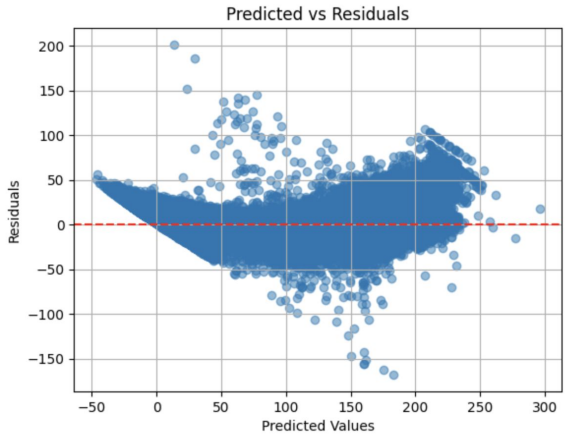
MSE: 230.61
RMSE: 15.18

4. Error Analysis



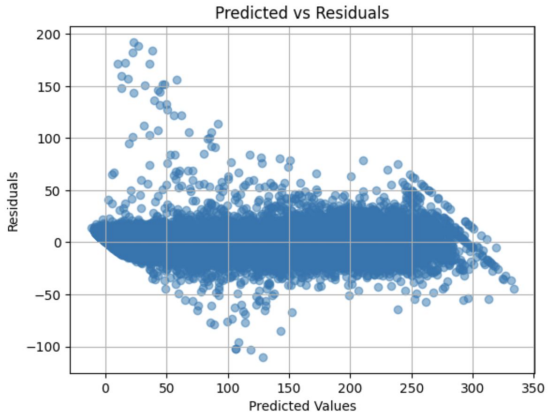
Model with 3 features:
y = w0 + w1 * 'Duration' + w2 * 'Heart_Rate' + w3 * 'Body_Temp'

MSE: 193.08
RMSE: 13.89



Model with all features:
y = w0 + w1 * 'Duration' + w2 * 'Heart_Rate' + w3 * 'Body_Temp'+ ...

MSE: 122.29
RMSE: 11.05



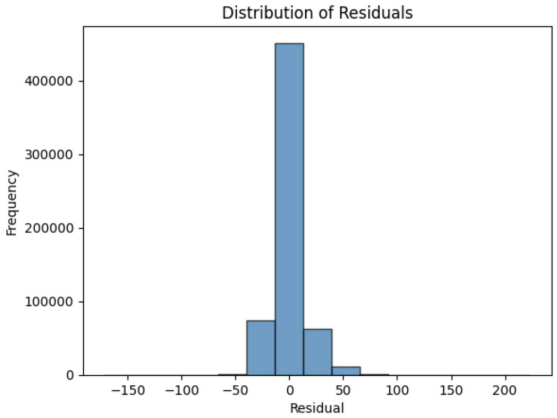
Polynomial Model with all features (degree=2): 7 numerical features -> 36 features

MSE: 19.35
RMSE: 4.39

Two quick ways to improve the model:

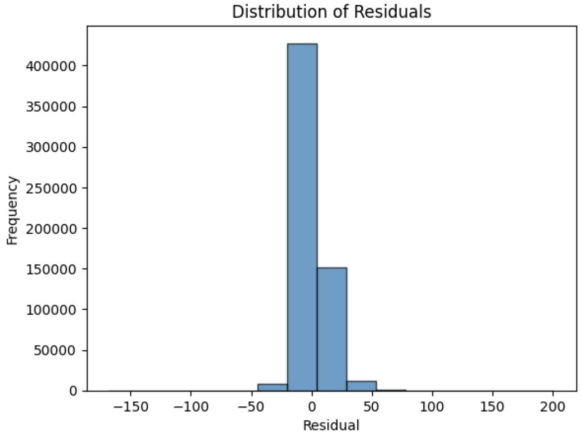
1. **Add more features or use all three original ones (the plot on the right)**
2. **Try polynomial regression or more complex model to catch nonlinearity**

4.1. Error Analysis. Histograms of Residuals



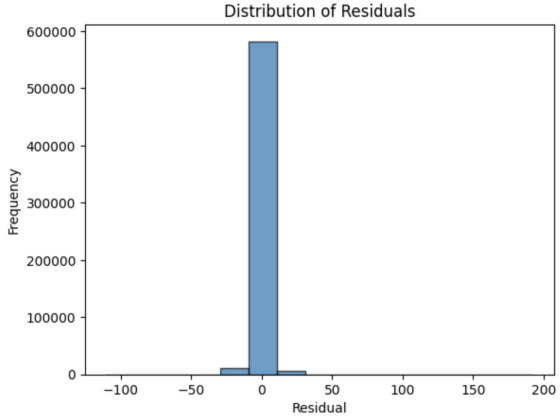
Model with 3 features:
y = w0 + w1 * 'Duration' + w2 * 'Heart_Rate' + w3 * 'Body_Temp'

MSE: 193.08
RMSE: 13.89



Model with all features:
y = w0 + w1 * 'Duration' + w2 * 'Heart_Rate' + w3 * 'Body_Temp'+ ...

MSE: 122.29
RMSE: 11.05

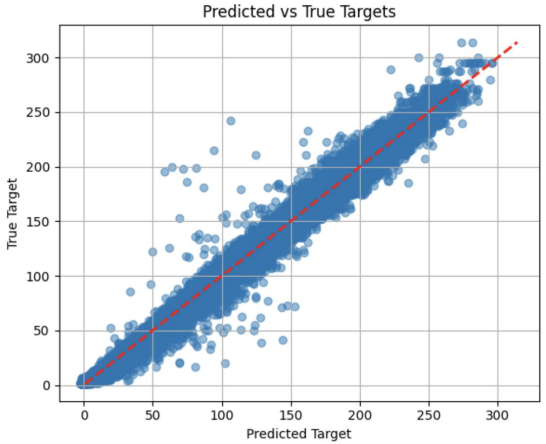


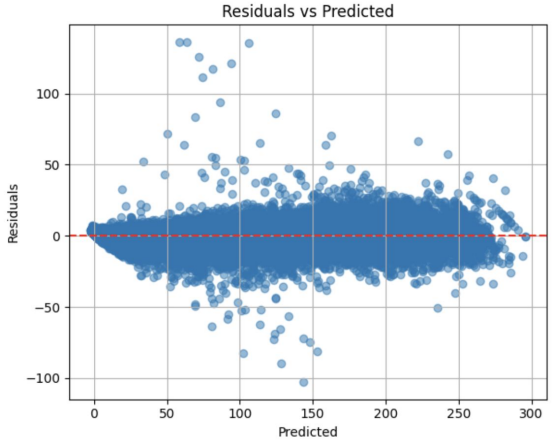
Polynomial Model with all features (degree=2): 7 numerical features -> 36 features

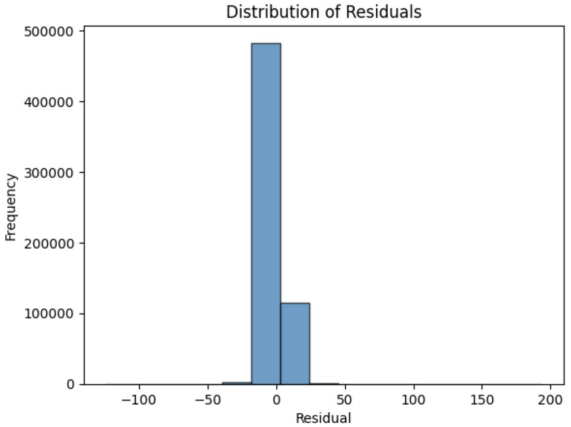
MSE: 19.35
RMSE: 4.39

5. Model Improvements

Hypothesis: Given the non-linear patterns observed in the Error Analysis and EDA for highly correlated features, gradient boosting will better capture these relationships than linear regression. All features were used for modeling.







model = GradientBoostingRegressor()

MSE: 22.57
RMSE: 4.75

Predicted values generally evenly distributed.
outliers with errors > +-100.
Conclusion: the model does not have a systematic error excluding errors at the ends of the range.

Mostly predicted-true errors are in the range -20 to +20

Conclusion: the model is generally balanced but sometimes underestimates real values.

6. Conclusions

Polynomial Model with all features (degree=2): 7 numerical features -> 36 features	**MSE: 19.35** **RMSE: 4.39**
Gradient Boosting Model	**MSE: 22.57** **RMSE: 4.75**

Gradient Boosting and Polynomial Regression give very close metrics and similar error graphs so based on metrics and error analysis they are approximately equal - model gives the same metrics and stable errors.

Next Steps:

1. We can go with Polynomial Regression or ->
2. Do cross-validation tests both models to check stability.
3. Clarify business requirements again what they looking for? Cost of deployment, interpretability, reliability.