A collection of gym equipment is shown against a grey background. In the foreground, there are several blue and teal dumbbells of various sizes, some labeled '2 KG', '2.5 KG', and '3 KG'. Behind them is a large blue exercise ball. To the right, a blue and black running shoe with a white sole is visible. The shoe has 'GEL-CUMULUS 11' and 'FluidRide' written on it. A black and white sock with 'BODY SCULPTURE' and 'SINCE 1965' is also visible. The text 'GYM EXERCISE CALORIES BURNED ANALYSIS' is overlaid in the center in white capital letters.

GYM EXERCISE CALORIES BURNED ANALYSIS

10 Features & 973 Rows

AGE

GENDER

AVG_BPM

Session
Duration
(HOURS)

BMI

WORKOUT
TYPE

FAT
PERCENTAGE

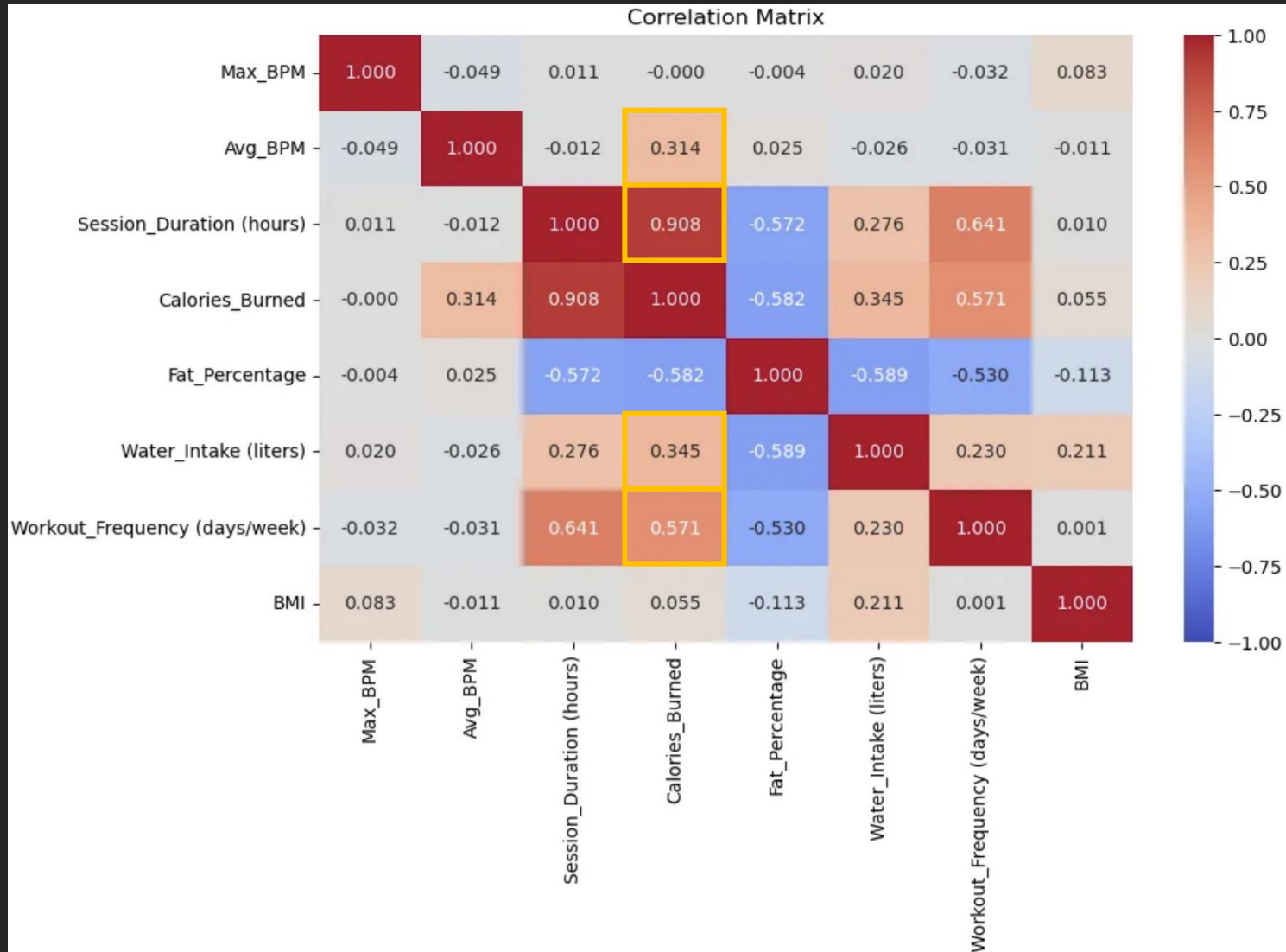
WATER INTAKE
(LITERS)

WORKOUT
FREQUENCY
(DAYS/WEEK)

CALORIES
BURNED

The background is a dark, textured surface with visible, expressive brushstrokes in shades of black and dark grey, creating a sense of movement and depth.

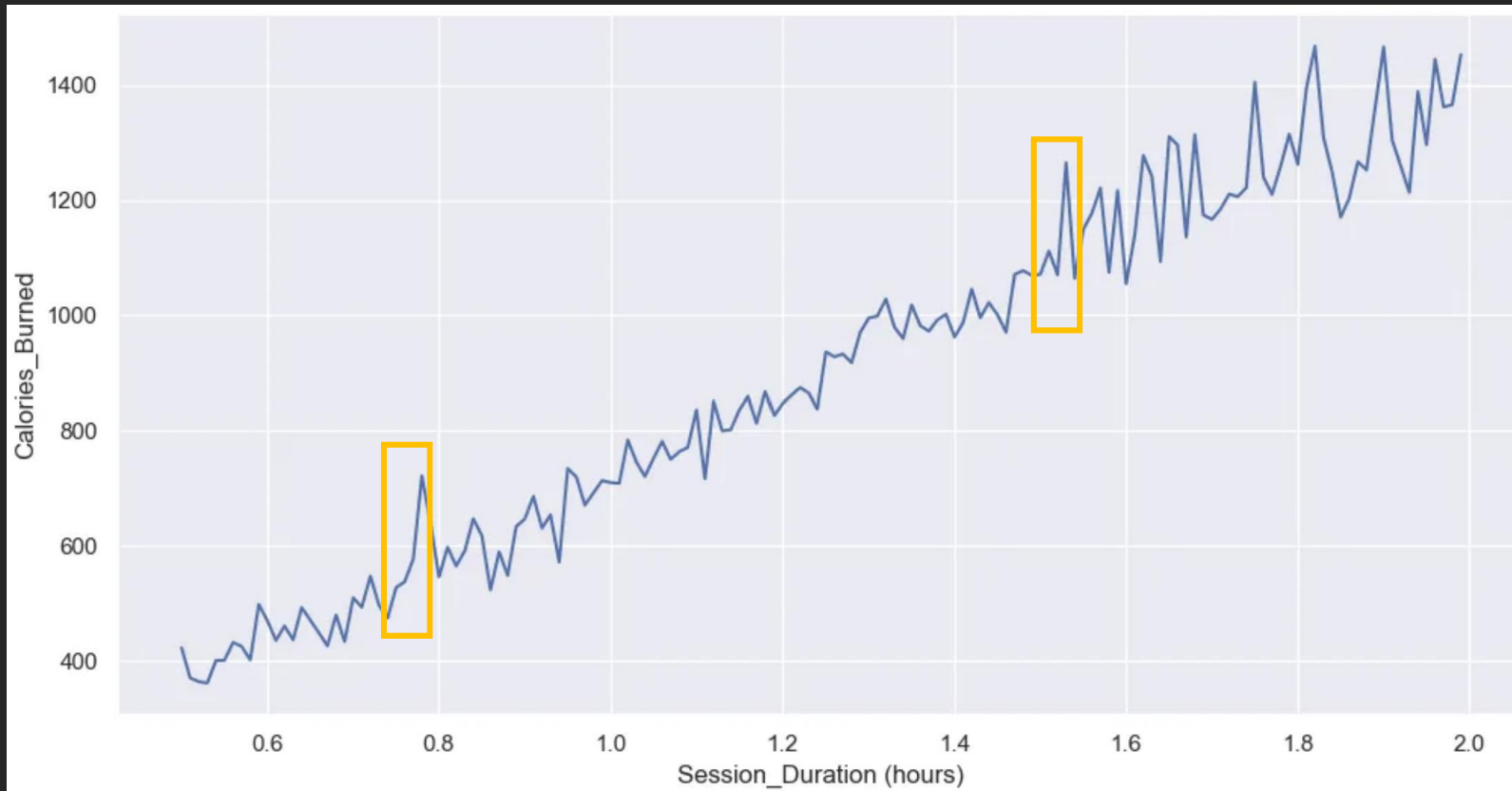
CORRELATIONS OF NUMERIC FEATURES



- It seems that **average BPM**, **session duration**, **water intake**, and **workout frequency** affect burning calories

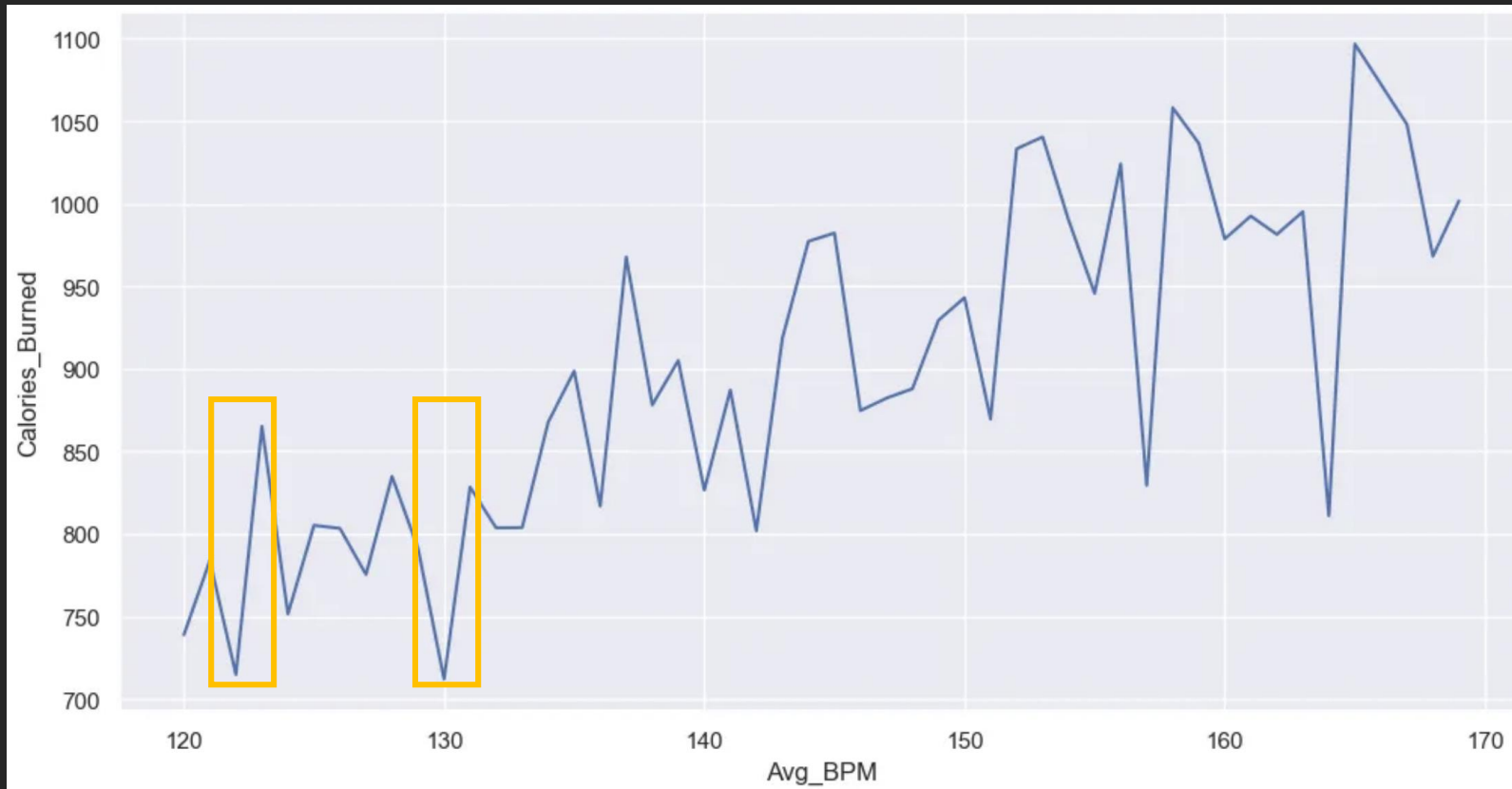
**Lets see the relations
in depth**

Session Duration and Calories Burned



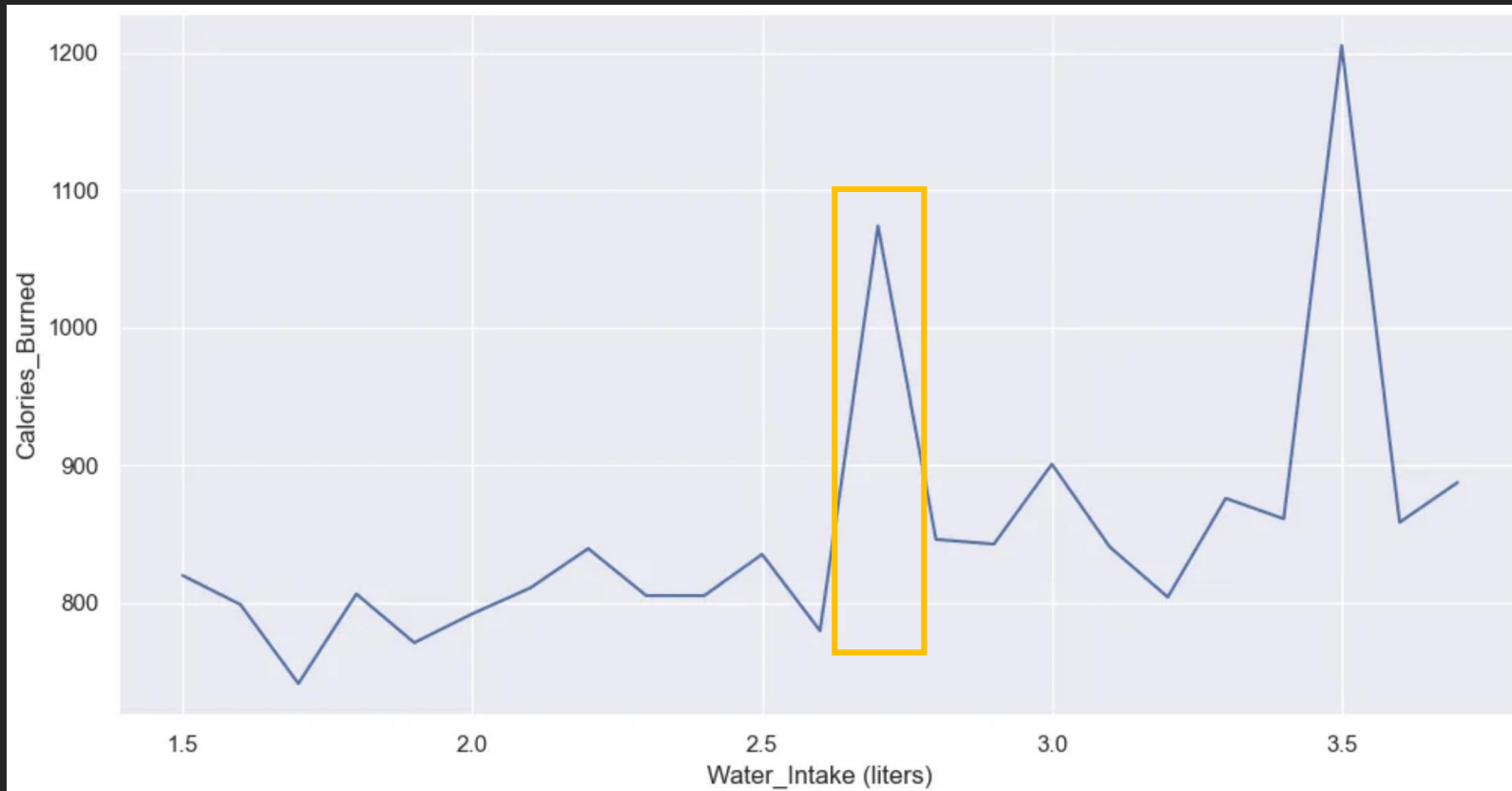
- The longer you do exercise, the more calories you can burn
- Keep eyes on the **sections** where the figures of calories burned drastically increase

Average BPM and Calories Burned



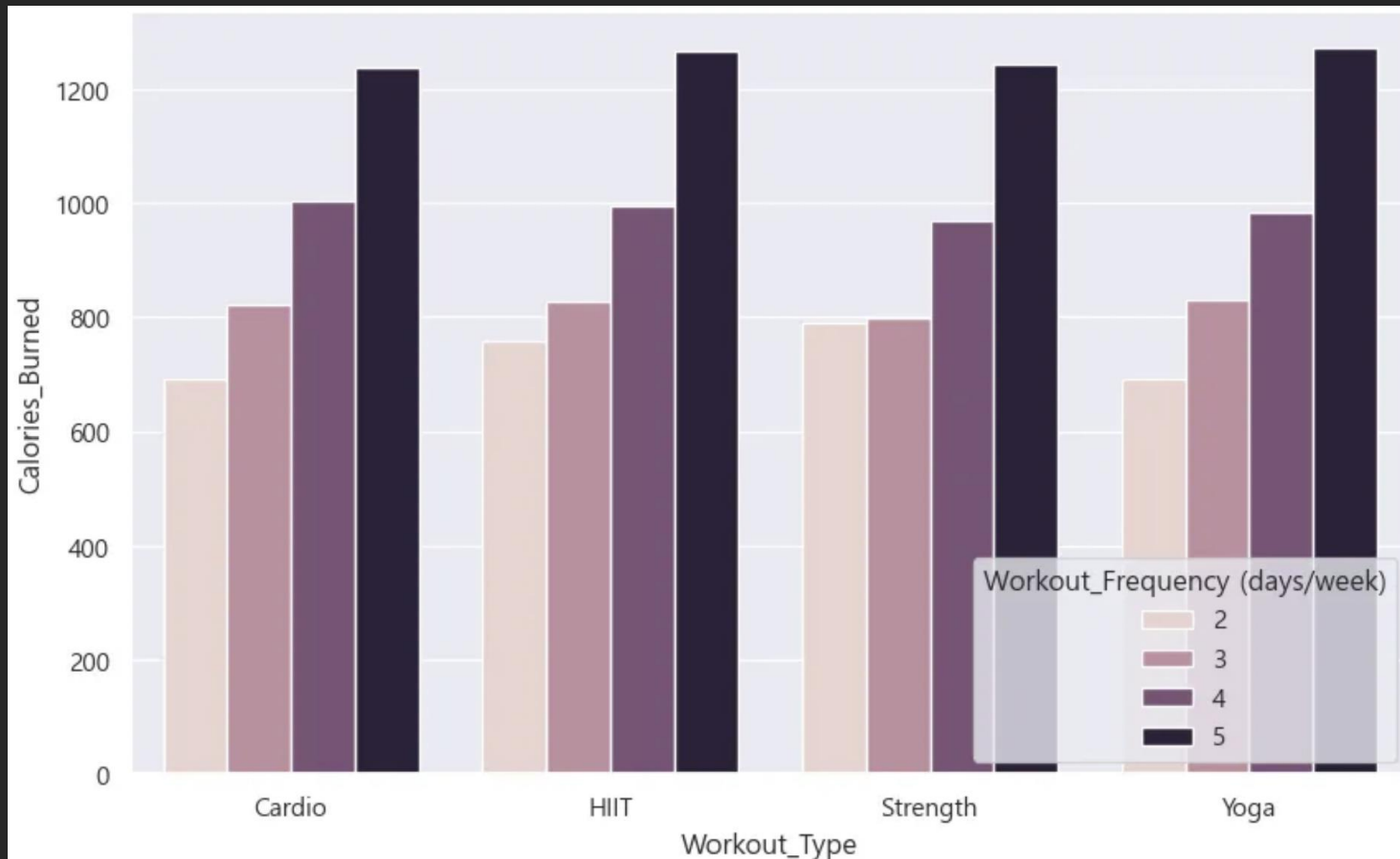
- Higher average BPM seems to burn more calories
- Keep eyes on the **sections** where the figures of calories burned drastically increase and decrease

Water Intake and Calories Burned



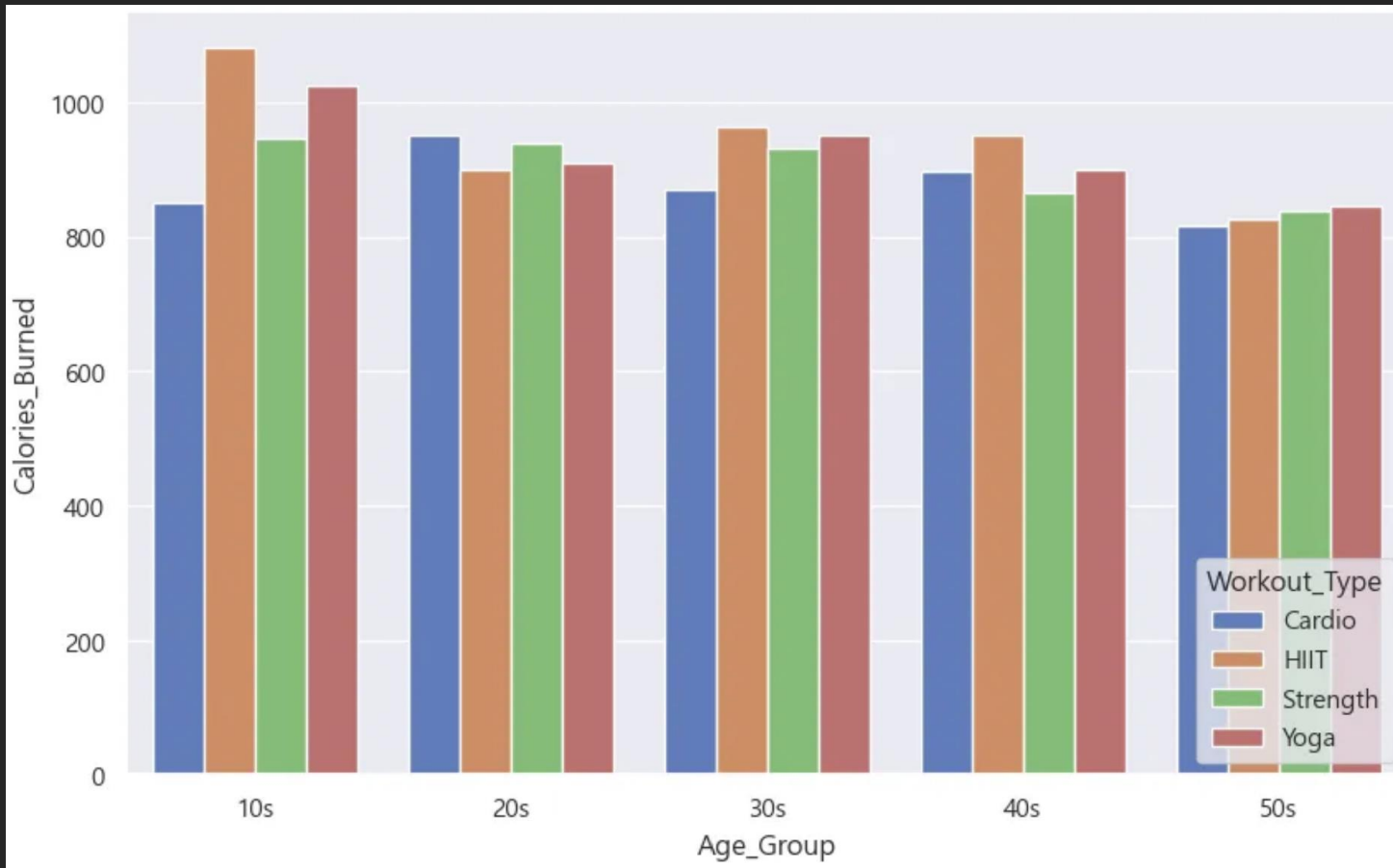
- The amount of water intake may have a positive impact on burning calories
- Keep eyes on the **sections** where the figures of calories burned drastically increase

Workout Type, Workout Frequency, and Calories Burned



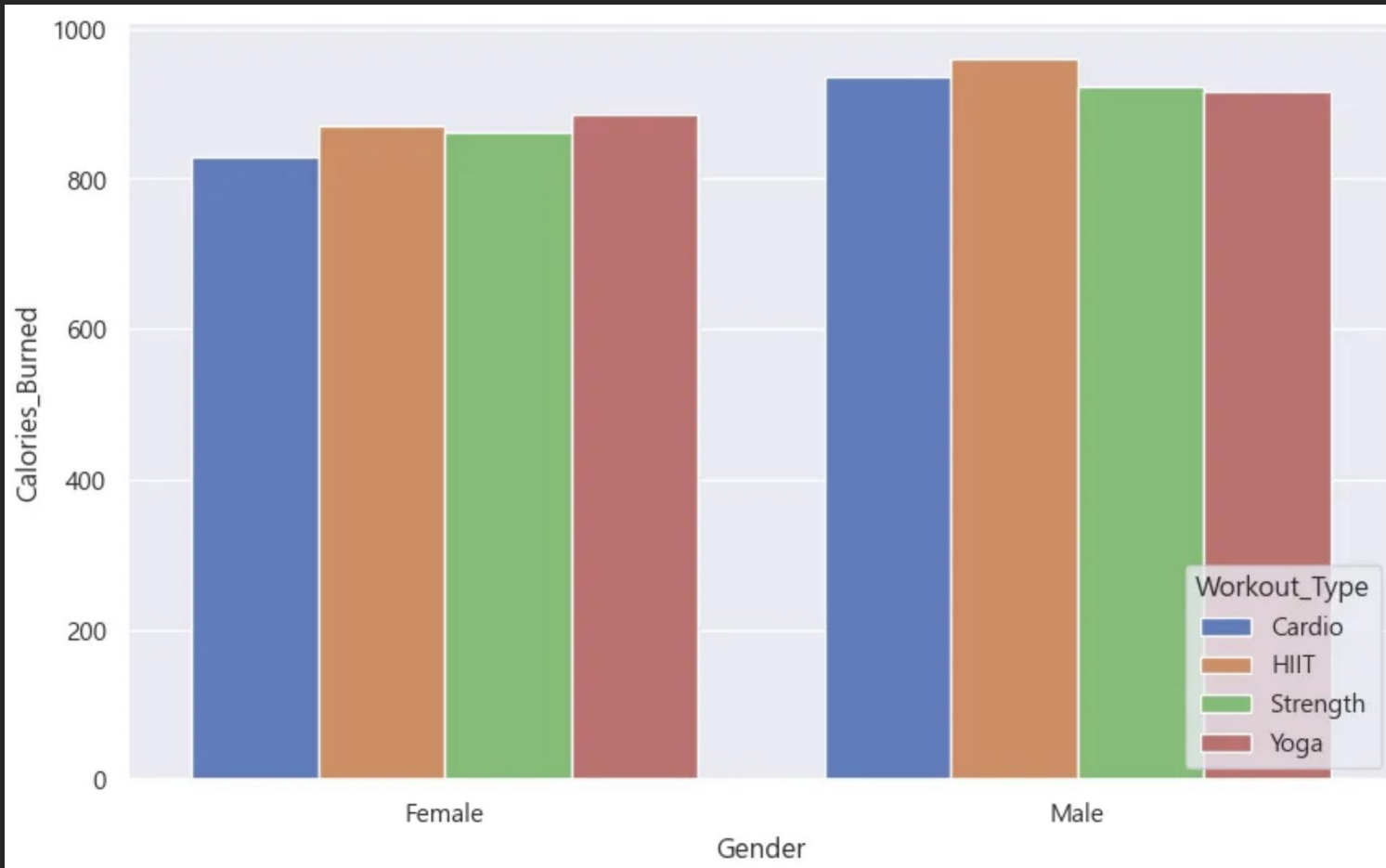
- Refer to the average calories burned depending on workout frequency to choose a workout type
- e.g.) If only **two days** are available for exercise per week, **strength** workout may be more effective to burn more calories

Age Group, Workout Type, and Calories Burned



- Refer to the average calories burned depending on age groups to choose a workout type
- e.g.) Regarding **teenagers**, **HIIT** may be the most effective workout type to burn calories

Gender, Workout Type, and Calories Burned



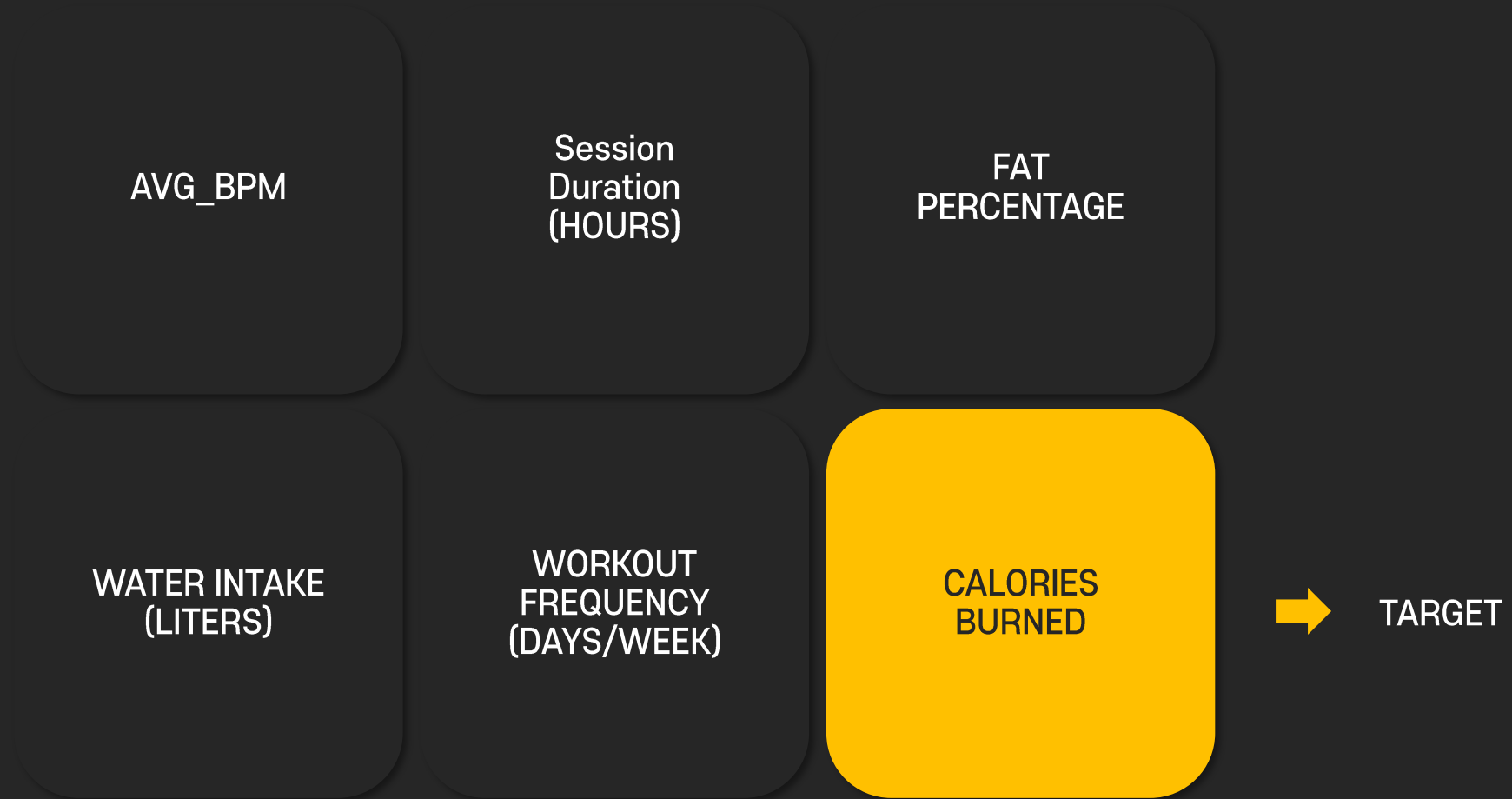
- Refer to the average calories burned depending on gender to choose a workout type
- e.g.) Regarding **female**, **Yoga** may be the most effective workout type to burn calories

The background is a dark, textured surface with abstract, dark grey and black brushstrokes or charcoal marks. The strokes are varied in direction and intensity, creating a sense of movement and depth. The overall tone is moody and artistic.

CALORIES BURNED PREDICTION MODEL

XGBOOST

Input Features and Target Feature



XGBOOST MODEL

```
import xgboost as xgb
from xgboost import plot_importance
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error

#Data preparation

X = df4[['Avg_BPM',
        'Session_Duration (hours)',
        'Fat_Percentage',
        'Water_Intake (liters)',
        'Workout_Frequency (days/week)', 'BMI']]

y = df4['Calories_Burned']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2,
                                                    random_state = 5)
```

```
#Model training and evaluation

xgb_reg = xgb.XGBRegressor(
    n_estimators = 50,
    max_depth = 5,
    gamma = 0,
    importance_type = 'gain',
    reg_lambda = 1,
    random_state = 100
)

xgb_reg.fit(X_train, y_train)

preds = xgb_reg.predict(X_test)

mae = mean_absolute_error(y_test, preds)

print(f'MAE: {mae}')
```

XGBOOST MODEL

```
#GridSearchCV -> best params: (max dpeth:5, n estimators: 50)

from sklearn.model_selection import GridSearchCV

params = {'n_estimators':[10, 50, 100], 'max_depth':[5, 10, 15]}

gridcv = GridSearchCV(xgb_reg, param_grid = params, cv = 3)

gridcv.fit(X_train, y_train)

print(gridcv.best_params_)

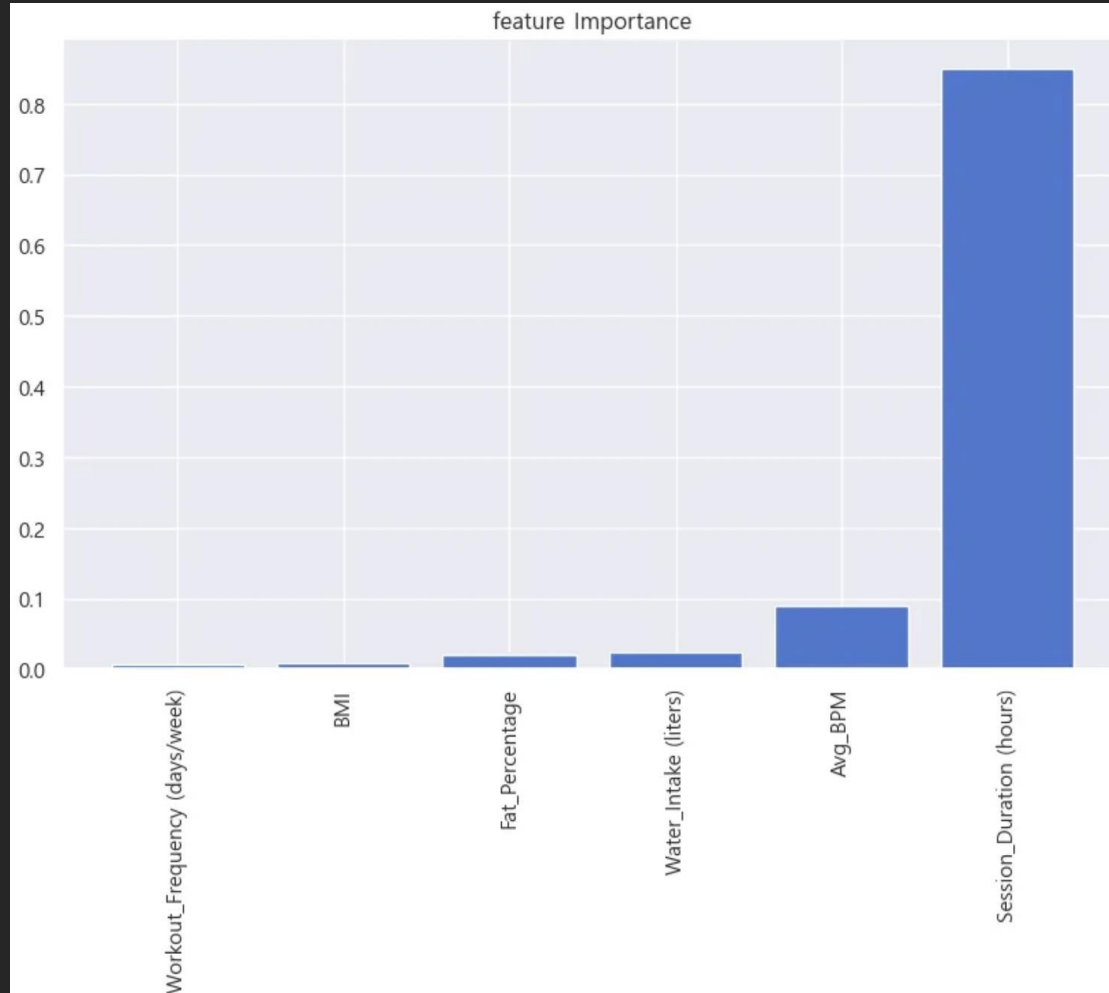
#Model test -> test target value: 1092

test_df = pd.DataFrame({'Avg_BPM' : [120],
                        'Session_Duration (hours)' : [2],
                        'Fat_Percentage' : [17],
                        'Water_Intake (liters)' : [1],
                        'Workout_Frequency (days/week)' : [3],
                        'BMI' : [30]})

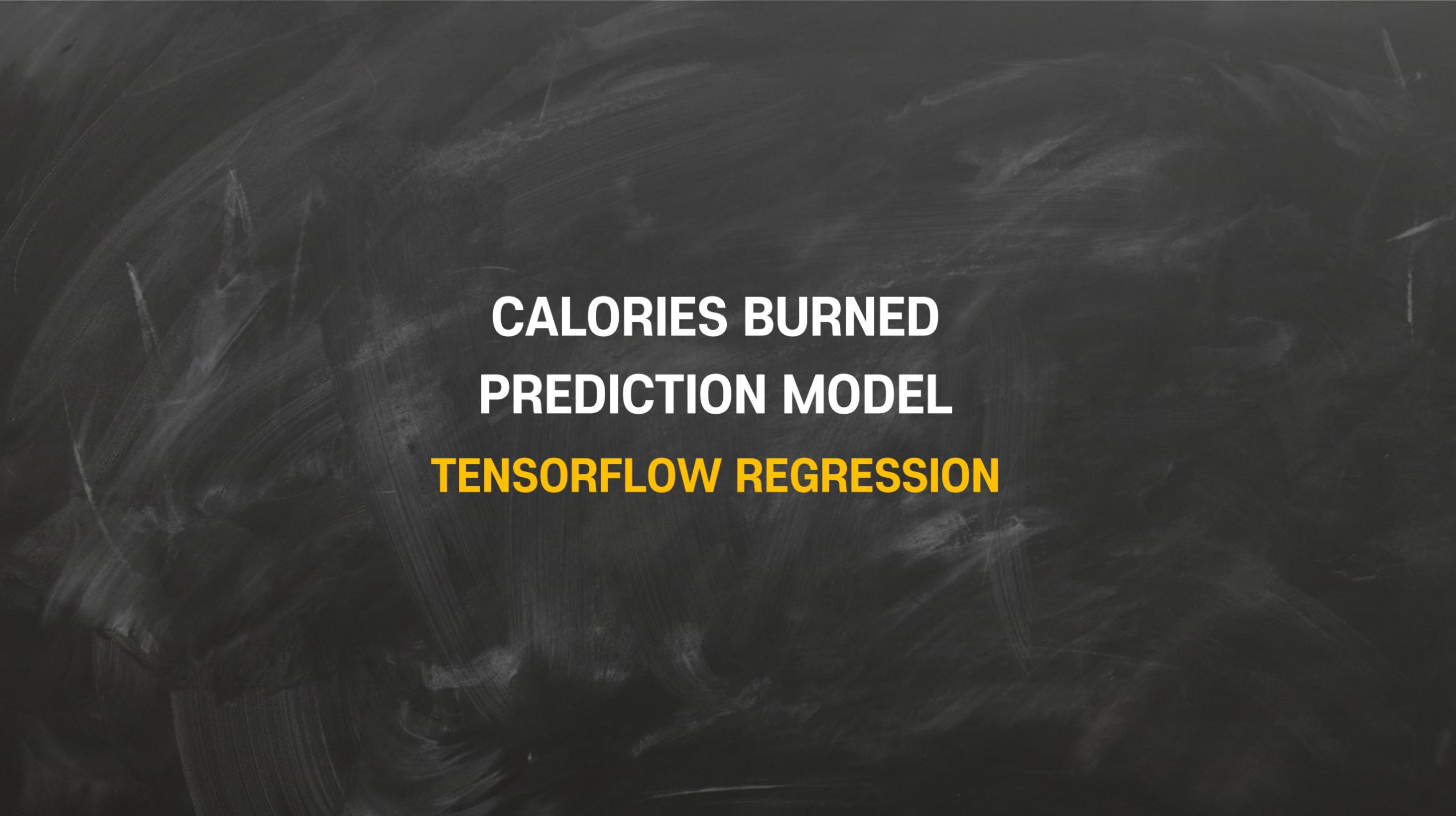
xgb_reg.predict(test_df)
```

MAE: 48.9

XGBOOST MODEL



- The most significant feature to predict the target is **session duration**

The background is a dark, textured surface with abstract, dark grey and black brushstrokes or charcoal marks. The strokes are varied in direction and intensity, creating a sense of movement and depth. The overall tone is moody and artistic.

CALORIES BURNED
PREDICTION MODEL
TENSORFLOW REGRESSION

Input Features and Target Feature

AVG_BPM

Session
Duration
(HOURS)

FAT
PERCENTAGE

WATER INTAKE
(LITERS)

WORKOUT
FREQUENCY
(DAYS/WEEK)

CALORIES
BURNED



TARGET

TENSORFLOW REGRESSION MODEL

```
#Data standardization

from sklearn.preprocessing import StandardScaler

standard_scaler = StandardScaler()

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2,
                                                    random_state = 5)

standard_scaler.fit(X_train)

X_train_standard = standard_scaler.transform(X_train)
X_test_standard = standard_scaler.transform(X_test)
```

```
#Create a model

tf.random.set_seed(42)

tf_model2 = tf.keras.Sequential([
    tf.keras.layers.Dense(100, activation = 'relu', name = 'layer_1'),
    tf.keras.layers.Dense(10, activation = 'relu', name = 'layer_2'),
    tf.keras.layers.Dense(1, name = 'output_layer')
])

tf_model2.compile(loss = tf.keras.losses.mae,
                  optimizer = tf.keras.optimizers.Adam(),
                  metrics = ['mae'])
```

TENSORFLOW REGRESSION MODEL

```
#Train the model -> loss: 45.1, mae: 45.1

train_history = tf_model2.fit(X_train_standard, y_train, epochs = 200)

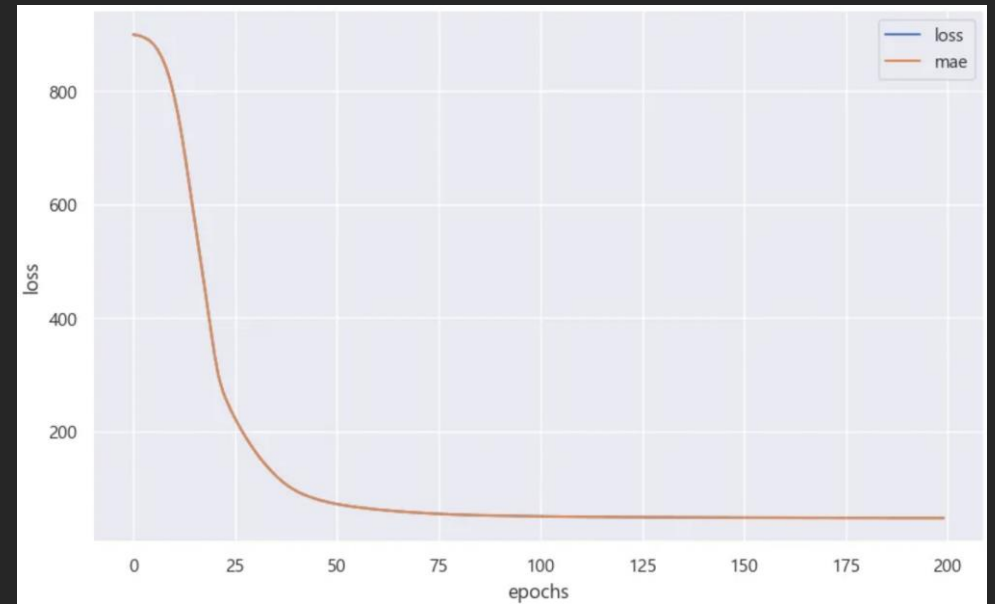
#Evaluate the model -> loss: 50.6, mae: 50.6

tf_model.evaluate(X_test, y_test)
```

```
#Visualize loss

pd.DataFrame(train_history.history).plot()
plt.ylabel('loss')
plt.xlabel('epochs')
```

MAE: 50.6



< Graph of Loss >