Powerlifter Deadlift Regression Analysis

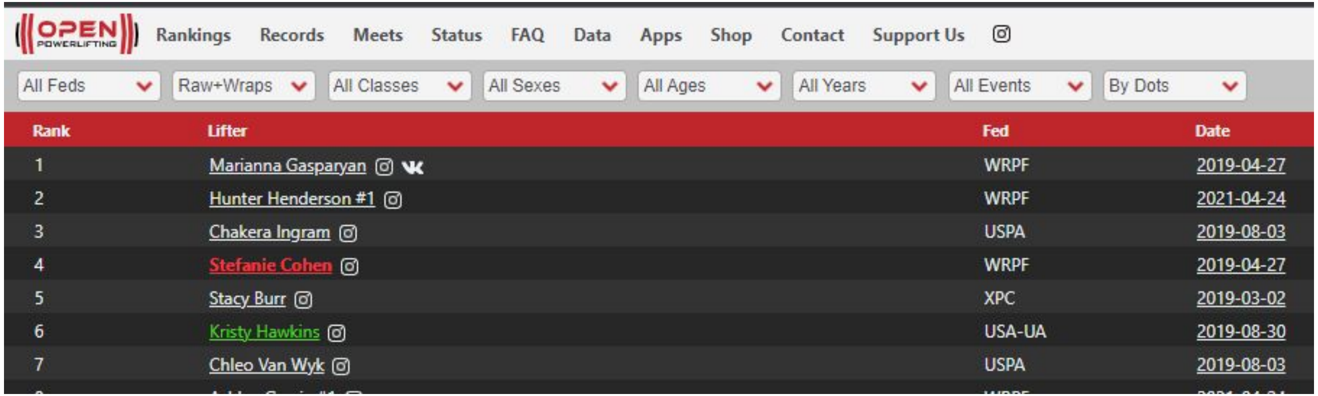
a Python Data Science project by Andrew Dettor

Problem

- How well can you **estimate** a powerlifter's deadlift performance?
- 1 rep max
- 3 attempts
- Squat, bench, deadlift
- Scope out competition ahead of time

Dataset

- openpowerlifting.org
- 1423354 samples
- 37 variables



Rank	Lifter	Fed	Date
1	Marianna Gasparyan	WRPF	2019-04-27
2	Hunter Henderson #1	WRPF	2021-04-24
3	Chakera Ingram	USPA	2019-08-03
4	Stefanie Cohen	WRPF	2019-04-27
5	Stacy Burr	XPC	2019-03-02
6	Kristy Hawkins	USA-UA	2019-08-30
7	Chleo Van Wyk	USPA	2019-08-03

Features

- As you would expect
- Age, sex, event, equipment used, division, bodyweight, federation, country of origin, etc
- Attempts for all three lifts (SBD)

Column	Non-Null Count
Name	1423354 non-null
Sex	1423354 non-null
Event	1423354 non-null
Equipment	1423354 non-null
Age	757527 non-null
AgeClass	786800 non-null
Division	1415176 non-null
BodyweightKg	1406622 non-null
WeightClassKg	1410042 non-null
Squat1Kg	337580 non-null
Squat2Kg	333349 non-null
Squat3Kg	323842 non-null
Squat4Kg	3696 non-null
Best3SquatKg	1031450 non-null
Bench1Kg	499779 non-null
Bench2Kg	493486 non-null
Bench3Kg	478485 non-null
Bench4Kg	9505 non-null
Best3BenchKg	1276181 non-null
Deadlift1Kg	363544 non-null
Deadlift2Kg	356023 non-null
Deadlift3Kg	339947 non-null
Deadlift4Kg	9246 non-null
Best3DeadliftKg	1081808 non-null
TotalKg	1313184 non-null
Place	1423354 non-null
Wilks	1304407 non-null
McCulloch	1304254 non-null
Glossbrenner	1304407 non-null
IPFPoints	1273286 non-null
Tested	1093892 non-null
Country	388884 non-null
Federation	1423354 non-null
Date	1423354 non-null
MeetCountry	1423354 non-null
MeetState	941545 non-null
MeetName	1423354 non-null

Data Cleaning

- Missing values
  - Many had >60% missing
  - Need to impute somehow
- Drop columns
  - Target leakage
- Drop rows
  - Event had all 3 lifts
  - Successful in all 3 lifts
- ~800,000 samples left

```
percent_missing(X)
```

```
Squat4Kg: 99.74%
Deadlift4Kg: 99.35%
Bench4Kg: 99.33%
Squat3Kg: 77.25%
Squat2Kg: 76.58%
Squat1Kg: 76.28%
Deadlift3Kg: 76.12%
Deadlift2Kg: 74.99%
Deadlift1Kg: 74.46%
Country: 72.68%
Bench3Kg: 66.38%
Bench2Kg: 65.33%
Bench1Kg: 64.89%
Age: 46.78%
```

Missing Value Imputation

- Categorical
  - Fill with “missing”
  - Country, division, meetstate, tested, sex, equipment
- Numerical
  - Iterative Imputer
  - Age, bodyweight

`sklearn.impute`.IterativeImputer¶

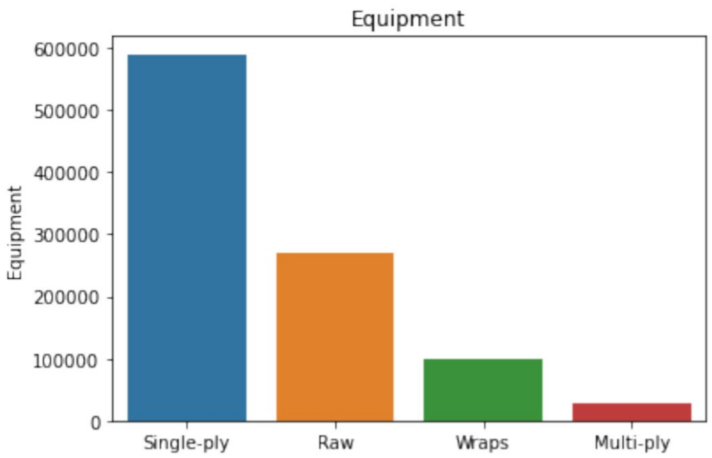
class sklearn.impute.IterativeImputer(*estimator=None, *, missing_values=nan, sample_posterior=False, max_iter=10, tol=0.001, n_nearest_features=None, initial_strategy='mean', imputation_order='ascending', skip_complete=False, min_value=- inf, max_value=inf, verbose=0, random_state=None, add_indicator=False*) [source]

Multivariate imputer that estimates each feature from all the others.

A strategy for imputing missing values by modeling each feature with missing values as a function of other features in a round-robin fashion.

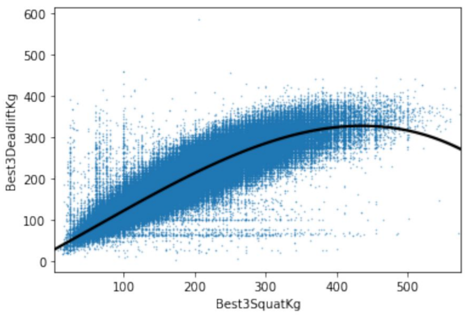
Exploratory Data Analysis (EDA)

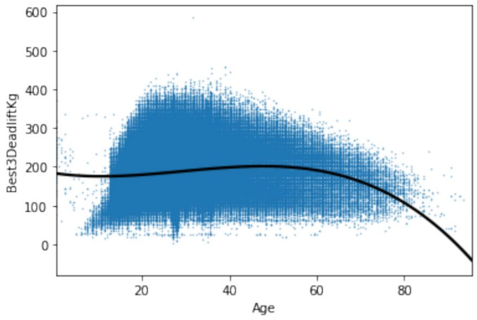
- Want to find interesting trends and connections
- Numerical
  - Correlations
  - Histograms
  - Line plots
- Categorical
  - Barplots
  - Boxplots
  - Pivot tables

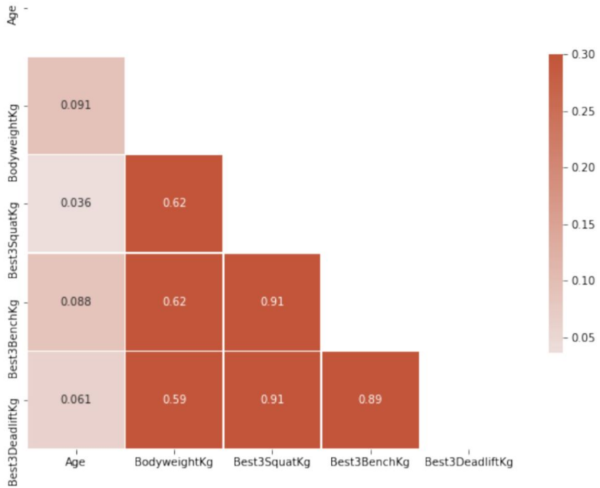


EDA - Numerical

- Correlations with body weight, best squat, and best deadlift
  - No surprise

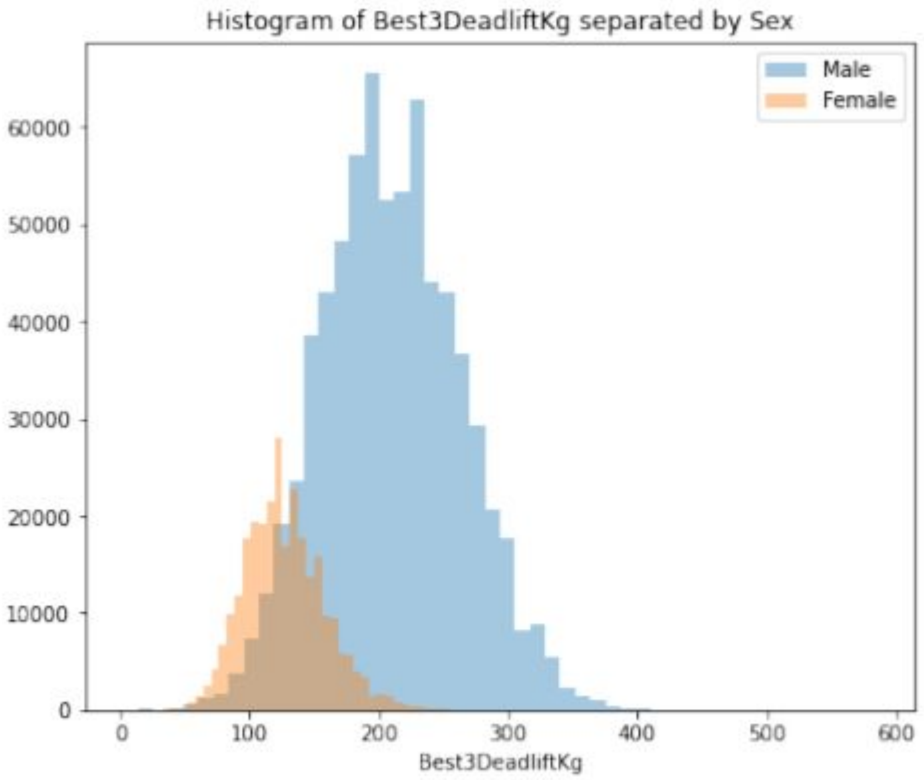






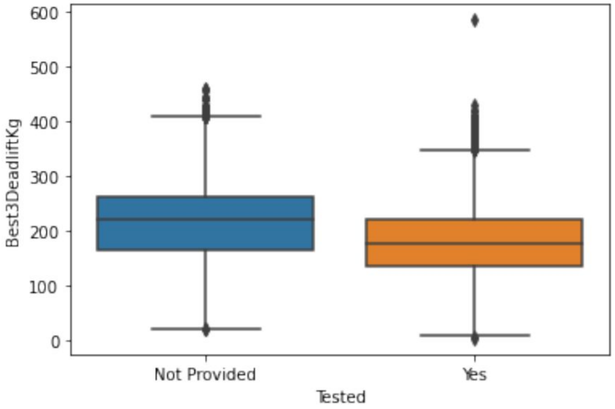
EDA - Numerical

- Sex differences
    - Also not surprising
    - Ratio



EDA - Categorical

- Performance enhancing drugs?
- Divisions?
- Country of origin?



Division	
Super Heavyweight	366.275
Open/Masters 40-44	365.000
Elite Pro Open	360.000
MM-2 RA	350.000
1974	350.000
	...
7-U	40.820
FR-M6	35.000
Ironman 8-9	34.020
Y 6-7	30.000
7-8	29.480

Country	
Yugoslavia	285.0
Bulgaria	275.0
Ghana	275.0
Swaziland	271.0
Central African Republic	270.0
	...
N.Ireland	155.0
Syria	152.5
Hong Kong	142.5
Nepal	140.0
Djibouti	120.0

Pivot Tables (average 1RM Deadlift for each categorical value)

Modelling - Plan

- Feature engineering
- Categorical encoding
- Preprocessing pipeline
- Model training
- Hyperparameter optimization
- Feature selection
- Interpretation

`sklearn.pipeline`.Pipeline

class sklearn.pipeline.Pipeline(*steps*, **, *memory=None*, *verbose=False*) [source]

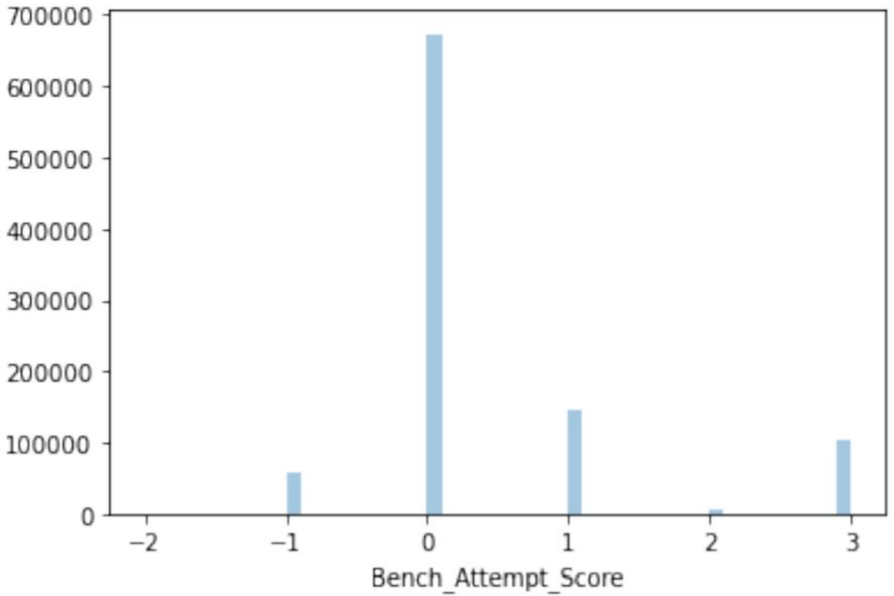
Pipeline of transforms with a final estimator.

Sequentially apply a list of transforms and a final estimator. Intermediate steps of the pipeline must be 'transforms', that is, they must implement `fit` and `transform` methods. The final estimator only needs to implement `fit`. The transformers in the pipeline can be cached using `memory` argument.

The purpose of the pipeline is to assemble several steps that can be cross-validated together while setting different parameters. For this, it enables setting parameters of the various steps using their names and the parameter name separated by a `'__'`, as in the example below. A step's estimator may be replaced entirely by setting the parameter with its name to another estimator, or a transformer removed by setting it to `'passthrough'` or `None`.

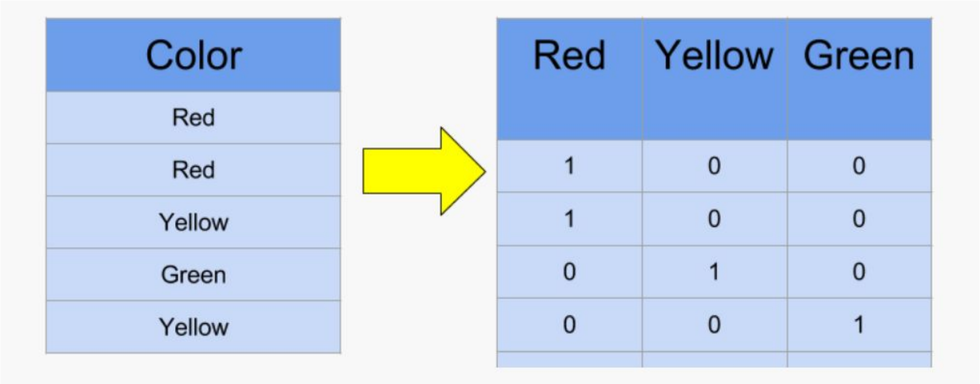
Feature Engineering

- Count fails and successes for each lift
- +1 for success, -1 for fail, 0 for unknown
- 6 features -> 3 features



Categorical Encoding

- Target Encoding
  - High cardinality
  - No target leakage
- One-Hot Encoding
  - Low cardinality



Color
Red
Red
Yellow
Green
Yellow

Red	Yellow	Green
1	0	0
1	0	0
0	1	0
0	0	1

Data Pipeline

- Scikit-learn API
- Custom Column Transformer
    - preprocessing
- Estimator
    - model
- Cross-validation
    - GroupShuffleSplit on Name
    - 5 splits

```
linear_regression_pipeline = Pipeline(
    steps=[
        ('columns', colTransformer),
        ('scale', StandardScaler()), # scales every column
        ('model cv', GridSearchCV(estimator=LinearRegression(),
                                param_grid={},
                                scoring="neg_mean_absolute_error",
                                cv = cv_indices_list))
    ],
    verbose=True
)
```

Modelling

- Performance
  - Best - XGBoost
  - Worst - Decision Tree
- Fit Time
  - Best - Ridge Regression
  - Worst - Random Forest
- Linear models do very well

- **Linear Regression**:
  15.16 MAE
  13.2 seconds to fit
- **Ridge Regression**:
  15.16 MAE
  9.6 seconds to fit
- **Decision Tree**:
  20.16 MAE
  75.9 seconds to fit
- **K Nearest Neighbors**:
  15.77 MAE
  12457.2 seconds to fit
- **XGBoost**:
  13.95 MAE
  324.4 seconds to fit
- **Random Forest**:
  14.78 MAE
  3345.3 seconds to fit
- **Linear Support Vector Machine**:
  15.12 MAE
  50.3 seconds to fit

Hyperparameter Optimization

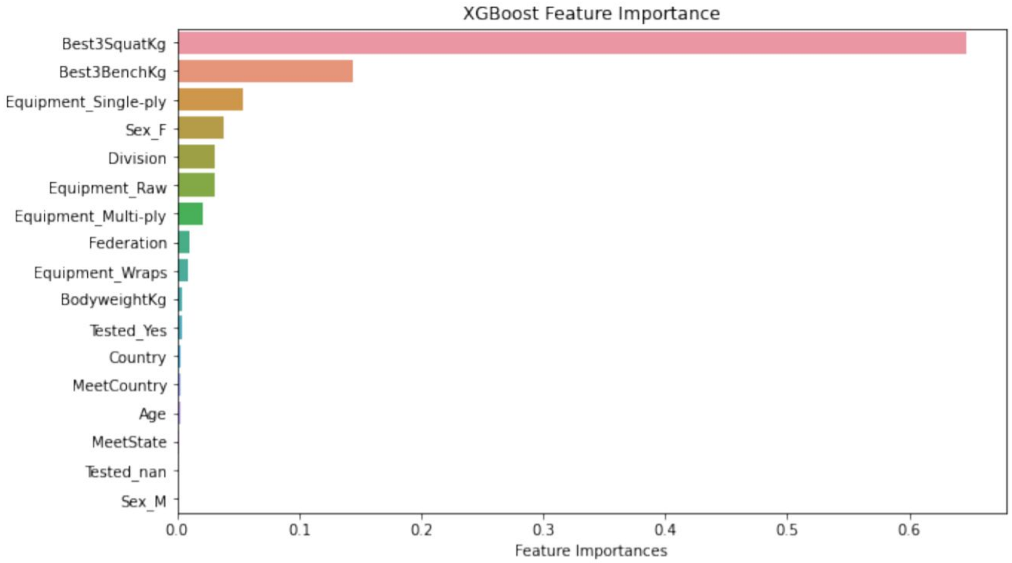
- Vary parameters
  - n_estimators
  - max_depth
  - Learning_rate
- Improve performance by .1
  - 13.95 -> 13.87
  - insignificant

```
xg_boost_regressor_pipeline_grid = Pipeline(
    steps=[
        ('columns', colTransformer),
        ('model grid search cv', GridSearchCV(estimator=XGBRegressor(),
                                              param_grid={'n_estimators': [50, 100, 150, 200],
                                                          'max_depth': [5, 6, 7, 8],
                                                          'learning_rate': [.1, .2, .3, .4]},
                                              scoring="neg_mean_absolute_error",
                                              cv = cv_indices_list))
    ],
    verbose=True
)

xg_boost_regressor_pipeline_grid.fit(Xtrain, ytrain)
```

Feature Selection

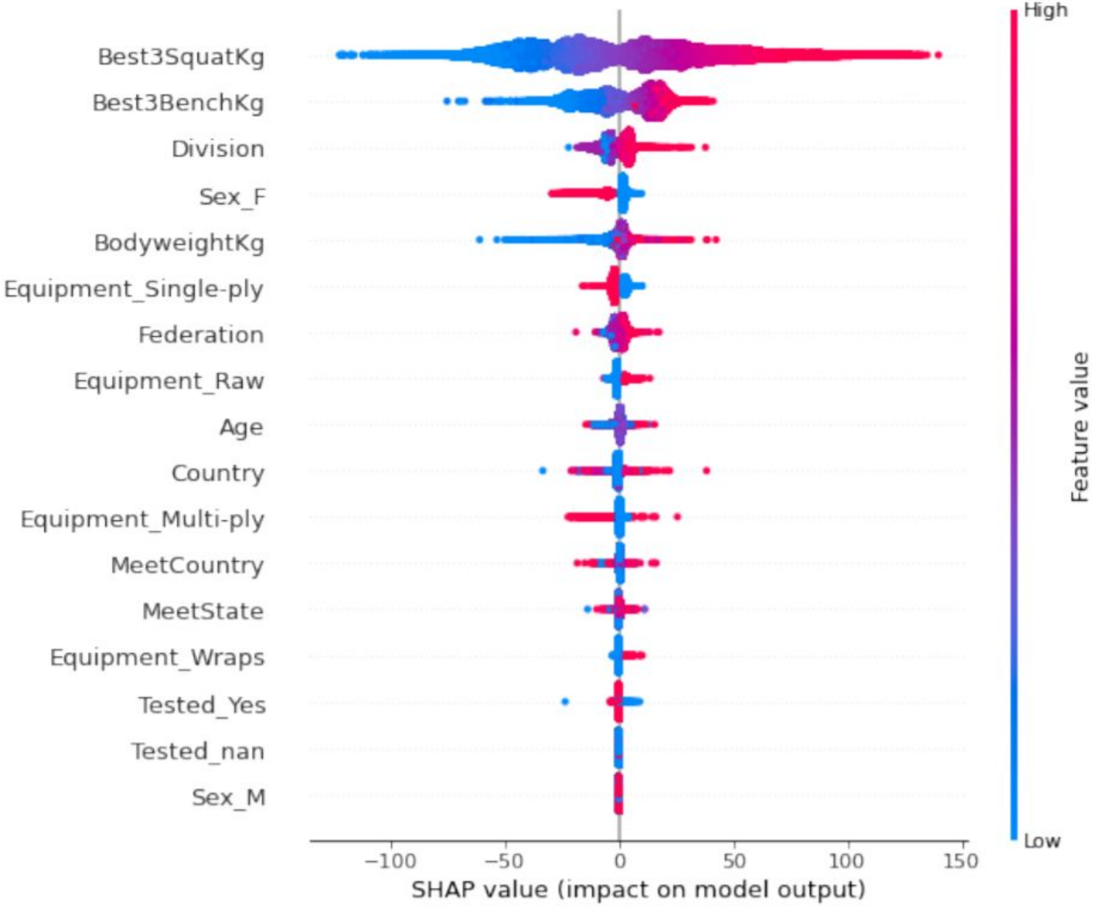
- Try for same performance with fewer features
  - 17 -> 10
- XGBoost Feature Importance
- Lasso Regression
- Permutation Importance
- All similar story
- 41% faster preprocessing
- 20% fit time
- 13.96 MAE (was 13.87)




```
removed_features = ["Sex_M", "Tested_Yes", "Tested_nan", "Equipment_Wraps",
                    "Age", "Country", "MeetCountry"]
```

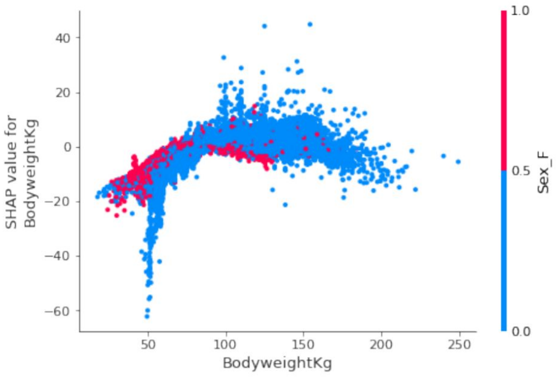
Interpretation

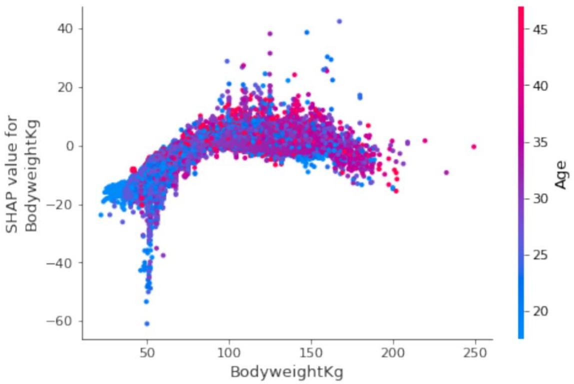
- Shapley Values
  - High shapley -> high deadlift
  - Model agnostic
  - Good visuals

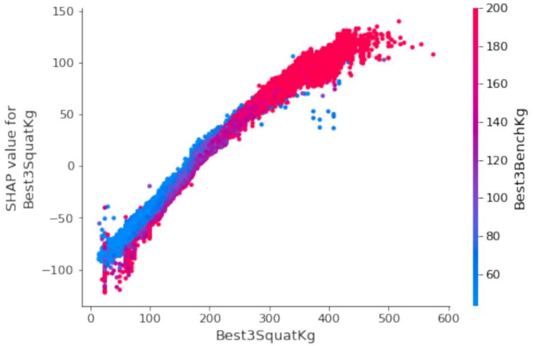


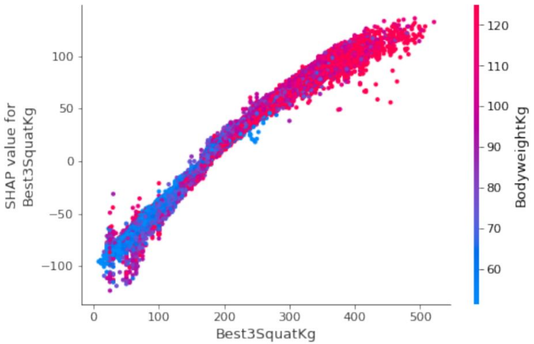
Interpretation

- Partial Dependence Plots
    - Deadlift vs 2 other features









Conclusion

- Other lifts matter the most
- Division, sex, and bodyweight are important, too
- Drug testing didn’t seem to matter much after all
- XGBoost worked best
    - ~14 kg average error
- Shapley values are awesome