Forecasting Climbing World Cup Performance

Ethan Brouwer, Romney Noel, Porter Bagley

January 3, 2021

1 Motivation

In recent years, the popularity of rock climbing has surged [1]. As more novices take to the climbing wall, more people are becoming interested in viewing rock climbing on a professional level. Each year, hundreds of professional athletes compete in the IFSC Climbing World Cup. A wide array of countries, ages, and body types are represented as well as varying years of experience. Competition climbing also has its nuances, and an athlete who excels outdoors might not demonstrate the same level of performance in the climbing gym.

Many professional sports, such as baseball and basketball, have had extensive analysis performed on them with the goals of identifying factors for success and forecasting winning teams. The detailed collection of data and the clear objective of sporting events make such problems well suited for forecasting using data analytics and machine learning techniques. Similar research in competition climbing has analyzed factors that impact performance, although their methods rely either on interviews with domain experts or experiments in closed, controlled settings [2, 3]. In this project, we analyze data on athletes and events in the IFSC Climbing World Cup. Our goal is to forecast the climbing world champions in the next 5 years. Throughout this paper, we focus on our results and analysis for the Men's Lead category. At the end, we run our model to get predictions in the other competition categories.

1.1 Overview of Data

Our data comes from the International Federation of Sport Climbing (IFSC). It was collected from the website digitalrock.de, a website maintained by Digital ROCK, a contractor for IFSC. Because the organization is contracted by the IFSC, we feel the data is trustworthy. This dataset is a good choice for our problem, because we are able to use the attributes of the athletes, and the trajectory of their career to make our predictions. We expect our analysis to reveal which of these features are most important in determining the performance of an athlete.

It is important to note that although in the end we want to forecast athletes' rank, the way the IFSC calculates the rank is somewhat ill-conditioned for our methods. This is because the rank is calculated based on an athletes' total score across all World Cup events, even if the athlete didn't attend several of them (which is quite common). This means that an athlete who places first place in all three events they attend could be ranked below an athlete who placed fifth place in six events. We feel that a more consistent metric for the performance of an athlete is the average points scored per event. Throughout our modeling efforts we use the average points per event as the target value, and we compare it to a ranking in the final step. It should also be noted that the

difference in the number of points awarded at the next highest rank increases drastically as the rank approaches 1st place.

1.2 Data Scraping

Athlete DataFrame:

Because of the way the website that we used to get our data was organized, to get athlete data we ended up running through about 200000 athlete ids (not all of which have athletes assigned) overnight with 50 separate workers running asynchronously to pull the data from their profile pages on the website. This dataset contains approximately 19,000 rows.

id	first_name	last_name	country	age	birth_year	height	weight
51001	Moritz	Simmet	GER	21	1999	nan	nan
51002	Martin	Eisensteger	GER	22	1997	nan	nan
51004	Vinzenz	Kreuzer	GER	24	1996	nan	nan
51007	Lukas	Achermann	SUI	17	2003	176 cm	66 kg
...	...	...	...	...	...	...	...

Rankings DataFrame:

To scrape the athletes' rankings, we used selenium to navigate between years on `https://www.digitalrock.de/icc_calendar.php` and found the IFSC or UIAA world ranking pages that had tables with athletes, their rank for the year, their total points for the year, what world cups they competed in, and their scores those years. This dataset contains approximately 8,600 rows.

id	rank	points	event	gender	year
8372	1	300	lead	MEN	2019
56609	2	256	lead	MEN	2019
5089	3	206	lead	MEN	2019
14023	4	195	lead	MEN	2019
...	...	...	...	...	...

World Cup DataFrame:

Finally, we realized that those ranking pages didn't include all the athlete's world cup scores for the year, so we scraped the final data from the website for each year from 1991-2019 so we knew all of an athlete's world cup scores for that year. This dataset contains approximately 38,600 rows.

athlete_id	rank	year	title	date	comp_id	cat_id	type
266	1	1991	uiaa worldcup - wien 1991	26 April 1991	85	1	lead
849	2	1991	uiaa worldcup - wien 1991	26 April 1991	85	1	lead
461	3	1991	uiaa worldcup - wien 1991	26 April 1991	85	1	lead
973	4	1991	uiaa worldcup - wien 1991	26 April 1991	85	1	lead
...	...	...	...	...	...	...	...

2 Data Cleaning / Feature Engineering

2.1 Cleaning

Our data was pretty clean to begin with, given that the website we used catered well to scraping with lots of clear tables and organization of data. We did however need to clean the following:

1. Drop the old age column because we already have the `birth_year` column that conveys information in a more understandable way
2. Setting `birth_year` to `NaN` for years that didn't make sense. Namely anything outside the range 1910-2017.
3. Convert the `birth_year` column into a new `age` column that actually represents how old they were at that event. Drop the `birth_year` column as it is simply a different form of the new `age` column.
4. Getting rid of units and converting height and weight into floats.
5. Dropping duplicate columns during merging process

[ ]:
```
# Cleaning the Athlete Data
athlete_df = pd.read_csv('data/athlete_data.csv', index_col='id')
clean_athletes = athlete_df.drop(columns='age')

bad_ages_mask = ~clean_athletes['birth_year'].isin(np.arange(1990-80, 2020-3))
bad_ages = clean_athletes[bad_ages_mask]['birth_year'].unique()
clean_athletes.loc[clean_athletes['birth_year'].isin(bad_ages)] = np.nan

clean_athletes['height'] = clean_athletes['height'].str.slice(0,-3).astype(float)
clean_athletes['weight'] = clean_athletes['weight'].str.slice(0,-2).astype(float)

# Merging the Athlete Data and the Ranking Data
ranking_df = pd.read_csv('data/rankings.csv')
ranked_athlete_ids = clean_rankings['ID'].unique()
ranked_athletes = athletes.loc[athletes.index.isin(ranked_athlete_ids)]
ranked_athletes['age'] = ranked_athletes['year'] - ranked_athletes['birth_year']
```

2.2 Feature Engineering

There were a few key features that engineered on our data that we believed could tell us a lot about the athlete's climbing ability. For the purpose of simplicity, we limit the following data to the Mens lead climbing discipline, but note that the same process can be applied to the other sets of data.

1. We first filter down to the rows with `event == 'lead'` and `gender == 'MEN'` to run our models on and remove the superfluous columns.
2. We add a `career_len` feature that tells us how many years an athlete has been competing since their first year. This feature applies to each year of competition and looks back their first year to compute it.
3. We add an `event_count` feature to get the amount of events an athlete went to that year. This is useful for calculating the "lag" columns later, but we eventually drop this in order to more easily predict further into the future.

4. We add an `avg_points` feature to each row of the athlete's average points in the world cups that year.
5. We generate "lag" columns. These columns are labeled as `t-x` where `x` is the number of years back that column represents. For example, `t-3` will hold the `avg_points` feature from 3 years before the given year. This allows us to train and test our data given that we know their performance for the previous K years. For our purposes, we chose $K = 7$ because that seemed to adequately span most climber's careers, giving us enough data for most climbers, without overfitting or ending up with too many `NaN`s in most rows.

[ ]:
```
# Filter our data to "MEN lead" ranked athletes
merged_ml = ranked_athletes[(ranked_athletes['event'] == 'lead') &␣
 ↪(ranked_athletes['gender'] == 'MEN')].copy()
merged_ml = merged_ml.drop(columns=['event', 'gender'])
merged_ml = merged_ml.reset_index(drop=True)

# Adding the career_len feature
athlete_ids = merged_ml['id'].unique()
experience = pd.DataFrame()
experience_list = []
for ID in athlete_ids:
    athlete = merged_ml[merged_ml['id'] == ID]
    # Get their first year
    starting_year = athlete['year'].min()
    # Calculate the career length
    athlete_experience = athlete['year'] - starting_year
    experience_list.append(athlete_experience)
# Add the feature
merged_ml['career_len'] = pd.concat(experience_list)

# Add event_count feature
men_lead_count_dfs = {}
for year in range(2019, 1990, -1):
    # Get the counts of events
    year_df = merged_ml[merged_ml['year'] == year]
    men_lead_count_dfs[year] = year_df['athlete_id'].value_counts()

all_counts = []
for i, row in merged_ml.iterrows():
    count_df = men_lead_count_dfs[row['year']]
    all_counts.append(count_df[row['id']])
merged_ml['event_count'] = all_counts

# Add avg_points feature
merged_ml['avg_points'] = merged_ml['Points'] / merged_ml['event_count']

# Add Lag Column Features
top_athletes = merged_ml['id'].unique()
```

```
K = 7 # Number of "Lag Columns" to generate
col_names = [f't-{k+1}' for k in range(K)]
lag_df = pd.DataFrame(columns=col_names)
for i, ID in enumerate(top_athletes):
    athlete = merged_ml[merged_ml['id'] == ID].copy()
    # Shift the avg_points over to create the lag
    lag_list = [athlete['avg_points'].shift(-(k+1)) for k in range(K)]
    lags = pd.concat(lag_list, axis=1)
    lags.columns = col_names
    # Add these lag columns to all the rest of the lag columns
    lag_df = pd.concat([lag_df, lags], axis=0)

# Add the lag columns to the original data
merged_ml = pd.concat([merged_ml, lag_df], axis=1)
```

2.3 Final Cleaning

We performed some small adjustments at the end of our cleaning and feature engineering, and before funning our models.

Imputing:

To impute our dataset and remove the `NaNs`, we did a few different things. 1. For height, weight, and age, we noticed that climbers from the same country tended to be similar in age and body type, so we imputed those values using the mean from the climbers country. 2. For the "lag" columns, we assumed that the climber had an avg_points of 0 in years where they didn't compete, so just replaced those `NaN` values with 0.

One-hot Encoding:

We one-hot encoded the country column so we can use it as a feature.

Train/Test Split:

We trained our models on the years 1990-2014 and then tested our model on the next 5 years (2015-2019). This allows us to test on a larger dataset which lowers the variance of our error.

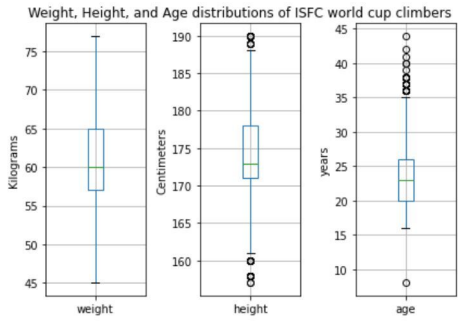
[ ]:
```
# `impute` is a longer function that compltes the imputing steps shown above
df = impute(merged_ml)
# Drop string and unimportant columns
df = df.drop(['id', 'last_name', 'first_name', 'points', 'rank', 'event_count'],␣
 ↪axis=1)
# One-hot encode the `country` column
df = pd.get_dummies(df)

# Get our test data
test = df[df['year'] >= 2015]
X_test = test.drop(['avg_points', 'year'], axis=1)
y_test = test['avg_points']
```

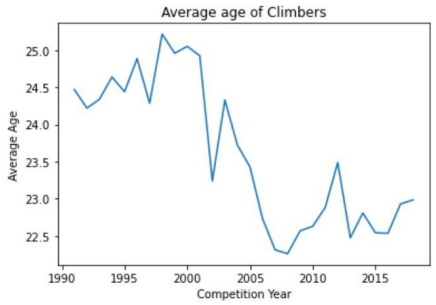
```
# Similarly, get our training data
train = df[df['year'] <= 2014]
X_train = train.drop(['avg_points', 'year'], axis=1)
y_train = train['avg_points']
```

Final Dataset:

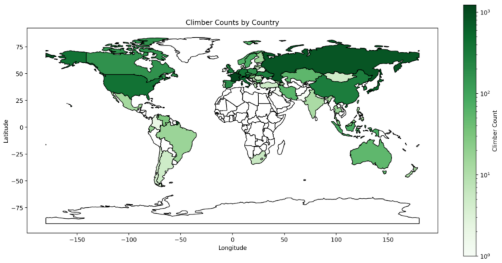
height	weight	age	career_len	avg_points	t-1	t-2	t-...	country_CZE	country_ESP	country_...
185	67	26	10	100	55	80	...	1	0	...
179.048	62.9524	29	1	42.6667	43	0	...	0	1	...
179.048	62.9524	20	16	34.3333	25.6667	63.5	...	0	0	...
179.048	62.9524	19	3	39	12	6.5	...	0	0	...
185	67	25	9	31.6667	66.5714	51.625	...	0	0	...
...	...	...	...	...	...	...	...	...	...	...

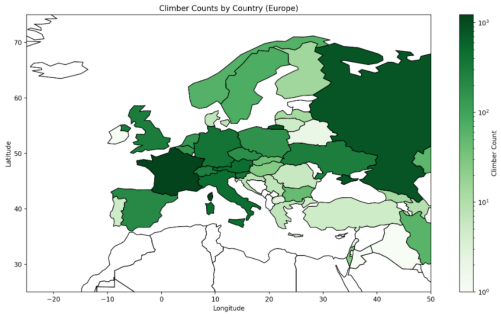
3 Data Visualization and Basic Analysis





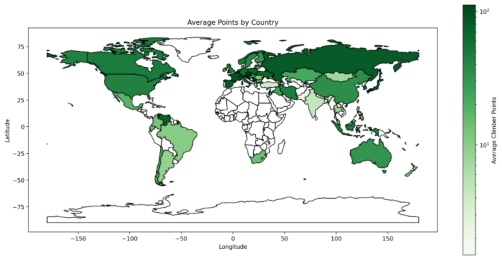
The two plots above show the distributions of physical characteristics of athletes. We can see there is a decent spread of heights, weights, and ages. The second plot shows an interesting trend where climber age has gone down on average since 1990. The variance in physical attributes suggests that this may be have a significant effect on our predictions.

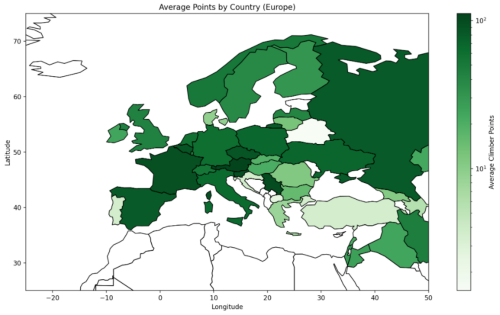




These four geographical visualizations show the distributions of climbers and climber performance across the world. Because much of the climbing community is centered in Europe, we provide a zoomed in figure for Europe. We see that there are a large number of climbers coming

out of Europe, especially in France. The highest scoring climbers are also coming from Europe. This suggests there may be some countries that are strong indicators for good climber performance.





4 Learning Algorithms and In-depth Analysis

4.1 Regression Models

To run our models and display our accuracy, we used the following code for each model. As a comparison, we used a naive baseline model which just predicted that athlete's scores would just be the same from the previous year.

Our metric for measuring the performance of our models was the Mean Squared Error (MSE) and the Mean Absolute Error (MAE). We choose to use the MSE because the MSE punishes large errors but is not so harsh on small errors. This was perfect for our purposes because we are attempting to regress near to our data without overfitting. The MAE was also included but not for optimization. Instead it was used only for interpretability. We show both the Mean Squared Error (MSE) and the Mean Absolute Error (MAE) for each of the models we ran.

[ ]:
```
def run_regression_model(model, param_grid, model_name):
    # Perform the Grid Search using all available cores
    grid = GridSearchCV(model, param_grid, n_jobs=-1,␣
↪scoring='neg_mean_squared_error').fit(X_train, y_train)
    # Predict based on our test data
    pred = grid.predict(X_test)
    # Print out the MSE of our predictions and the best params found with the␣
↪grid search
    print("MSE Value:", mean_squared_error(pred, y_test))
    print("MAE Value:", mean_absolute_error(pred, y_test))
    print("Best Params:", grid.best_params_)
```

Our best model by the MSE was Linear Regression. Despite the high MSE of XGBoost, however, it had a surprisingly low MAE, leading us to believe that it got many close predictions with a few predictions very far away, as MSE heavily penalizes being farther away from the correct value.

We also pulled the feature importances from our Random Forest Regressor, which performed better than our baseline. We show the 10 most important features below:

As we thought, our lag columns were more important than most features. Specifically, the previ-

Model Name	Best Parameters	MSE	MAE
Baseline Model	N/A	195.27	9.65
Linear Regression	Default	155.18	9.21
Linear Regression (Lasso Regularization)	{'alpha': 0.1}	159.64	9.32
Linear Regression (Ridge Regularization)	{'alpha': 1000.0}	155.5	9.08
Linear Regression (Elastic Net Regularization)	{'alpha': 0.1, 'l1_ratio': 0.001}	159.65	9.32
Decision Tree Regressor	{'max_depth': 10, 'max_leaf_nodes': 10, 'min_samples_leaf': 1, 'min_samples_split': 2}	168.47	9.54
Random Forest Regressor	{'max_depth': 5, 'max_features': 'auto', 'min_samples_leaf': 10, 'n_estimators': 50}	169.76	9.66
Gradient Boosting Regressor	{'learning_rate': 0.1, 'max_depth': 2, 'max_features': 'auto', 'min_samples_leaf': 10, 'n_estimators': 50}	170.11	9.65
XGBoost	{'alpha': 0, 'eta': 0.1, 'gamma': 0, 'lambda': 0, 'max_depth': 1}	234.38	8.66

Feature	Importance
t-1	0.828737
t-2	0.087144
t-3	0.019983
age	0.013638
weight	0.010308
t-4	0.009976
country_FRA	0.007835
career_len	0.007252
height	0.006860
t-5	0.001890

ous year's score proved to be a very important indicator of their performance the next year.

Womens Lead:

We also ran our code on the womens lead discipline with our Linear Regression model and found similar results with an MSE of 162.35 and and MAE of 9.10.

4.2 Predicting 5 Years into the Future

Returning to our original research question, we hoped to be able to accurately predict the best climbers 5 years into the future. To do this, we trained our model as before on data from years 1990-2015. Then we predicted the next year's averages. Then we shifted our lag columns down, letting `avg_points` be the predicted values, `t-1` the previous year's values, and so forth. Then, we predict the next year's data and so on until we reach 2019. Here are our predictions, and their accuracy. We also include a figure showing the accuracy (MSE) for each of the years up to and including 2019.

First Name	Last Name	Predicted Avg	Actual Avg	Difference	Predicted Rank	Actual
Jakob	Schubert	63.0651	44	19.0651	1	14
Adam	Ondra	57.0388	100	42.9612	2	1
Domen	Skofic	44.3178	36.6667	7.65118	3	22
Stefano	Ghisolfi	43.8053	31.6667	12.1386	4	5
Sean	McColl	42.5345	34.3333	8.20118	5	3

4.3 Models we chose not to use

Because our target value was continuous, we only used regression methods, rather than using classification methods. We also largely avoided unsupervised methods. During our exploration phase, we ran t-SNE, but were unable to observe any meaningful clusters. We determined that the problem we were most interested in solving was the supervised problem.

5 Ethical Implications

With our results, it is important to make several ethical considerations. First, our model includes country of origin as a feature. Although certain countries were identified as important features, we are in no way suggesting that an individual athlete should be assumed to be a good/bad climber based on the country they are from.

Second, if our model became widely used, it could produce a feedback loop between sponsorship and athlete performance. The industry of professional climbing is not yet at a point where successful athletes make a lot of money, therefore sponsorship may make a significant difference on how much time a professional athlete is able to devote to training, and what gear/resources are available to them . If sponsors could forecast with high reliability who is likely to perform best next year, they would want to sponsor those athletes. Generous sponsorship would allow those athletes to train more and perform better the year following, and consequently receive more sponsorships. This would negatively affect those athletes who may have received sponsorship without our model being considered. Our model will currently not be made available publicly, and if it were eventually, we would ensure that it was maintainable, and could be deactivated if needed.

6 Conclusion

In conclusion, we were successful in outperforming the baseline model in predicting the average points per event of athletes. Our model improved the MSE from 195.7 to 155.5 (-40.2). Furthermore, our analysis showed that the biggest predictors of an athletes' performance are their performance in the previous three years, along with their age and weight. However, as we anticipated,

our model did not perform as well when predicting the ranks of athletes. This is a result of athletes skipping events. While our model may not be able to predict the winners of the IFSC World Cup in future years, we feel that it does a decent job of predicting which climbers will perform the best on average. That is to say that if each athlete attended all the events, we feel our model would be able to predict World Cup winners with much higher confidence.

References

[1] Dwyer, K. (2019, November 14). "Social Climbing Has a Whole New Meaning". *The New York Times*. https://www.nytimes.com/2019/11/14/style/bouldering.html

[2] Sanchez Xavier, Torregrossa M., Woodman T., Jones G., Llewellyn D. J. (2019). "Identification of Parameters That Predict Sport Climbing Performance". *Frontiers in Psychology* 10.3389/fpsyg.2019.01294.

[3] Arazi, Hamid, et al. "Profiling and predicting performance of indoor rock climbers." *Revista Brasileira de Cineantropometria & Desempenho Humano* 20.1 (2018): 82-94.