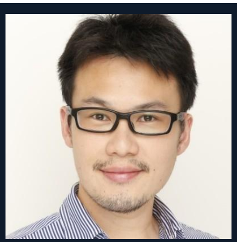
Predicting the Super Bowl LVI Winner

Mychael Gray

Data Science Intensive Capstone Project, November 1st 2021 Cohort

Springboard

Thanks to Springboard mentors



A Wang Ang, Principal Data Scientist

Michelle Jorgensen, Springboard Student Advisor

The Problem

According to data from the National Retail Federation (NRF), total spending for food, drinks, apparel, decorations, and other purchases for the day is expected to reach…

Americans spent:

16+ billions

or…

$85.36 per person.

Are there more important factors and qualities of a NFL Super Bowl team and can all of these factors be used to predict a winner accurately, from the 2022 season?

Who might care?

Investors

Businesses

DIAMOND
DEALER DIRECT

Buyers/Consumers

…and many more…

What factors might affect prices?

- **Prices: set by sellers or wholesalers**
- **Carat or weight**
- **Cut or quality**
- **Clarity**
- **Color**

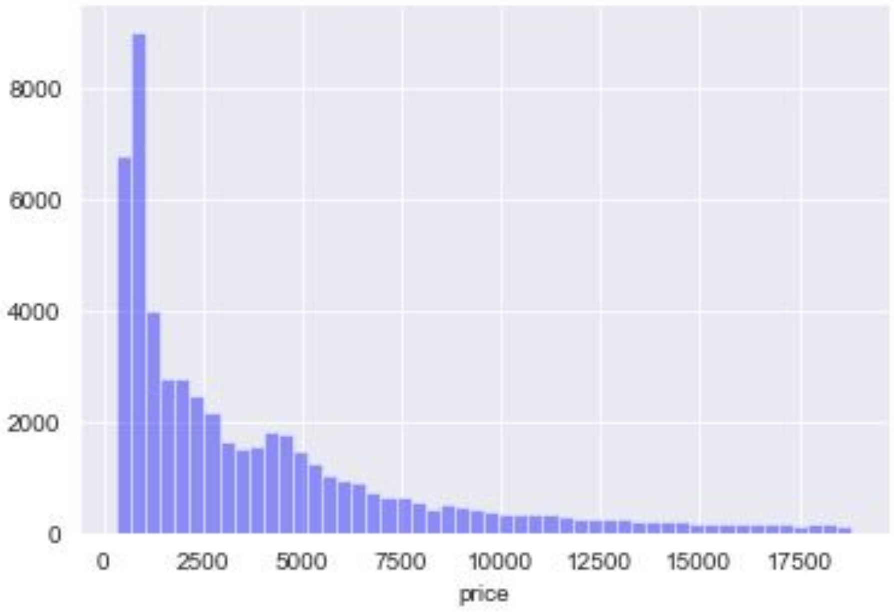
kaggle

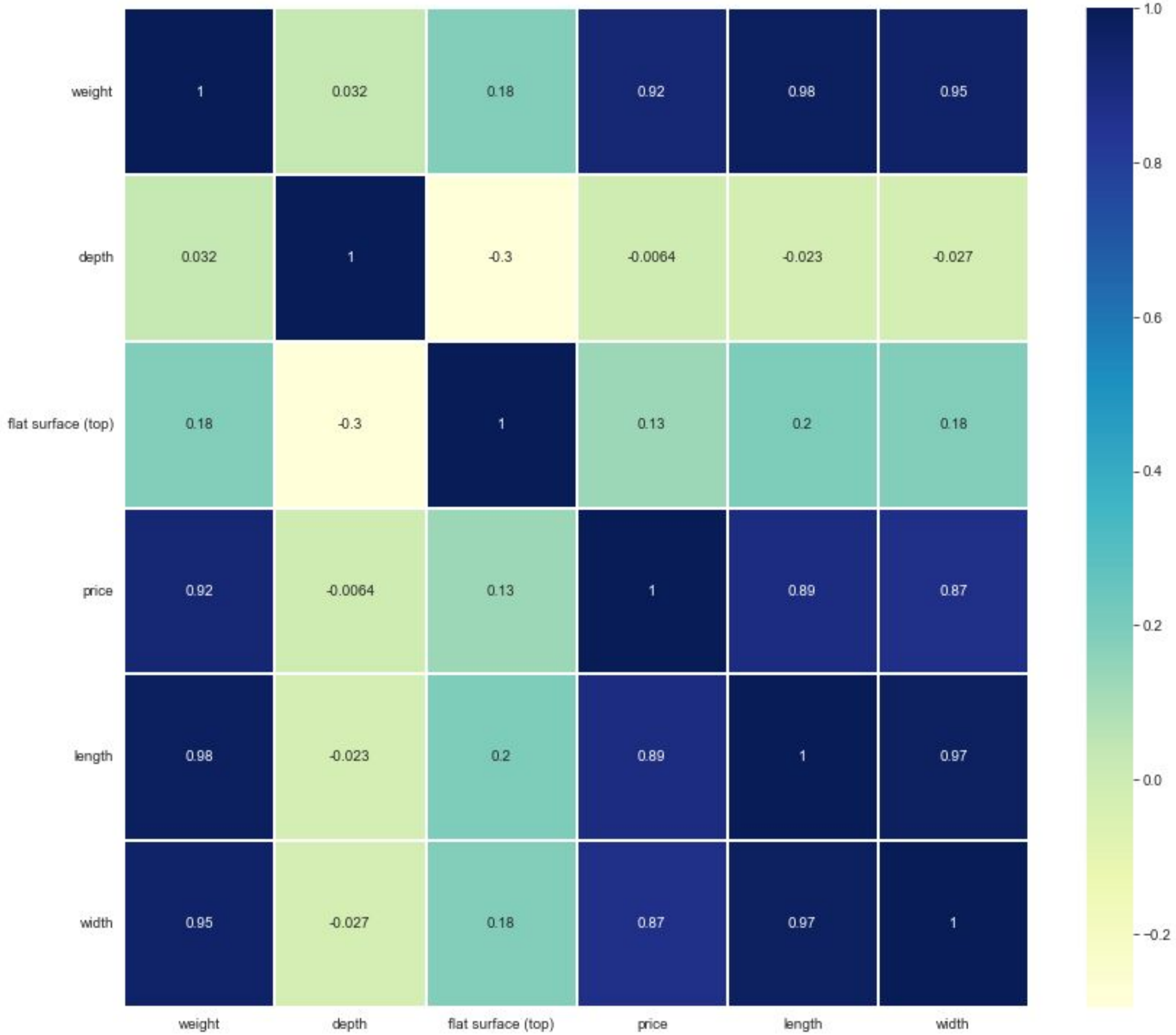
Data Source

- **Technical Factors:**
- **width**
- **Table**
- **length**
- **depth**

Industry

Some Historical Indicators





Initially when exploring the data, we could see that the lowest prices start at 326.00 in US dollars with 2 observations of that amount and can raise all the way up to 18823.00 in US dollars with 1 observation at that amount.

Modeling Overview

Type: Supervised learning

Regression Model: Predicting Price

No missing data & feature names updated

Tools: Python 3, scikit learn, and matplotlib

Modeling Steps

Data pre-processing steps:

1. Check data types
2. Create dummy or indicator features for categorical variables
3. Data splitting into training and test sets **(80% / 20%)**
4. Standardize the magnitude of numeric features using a scaler

Optional: Cross validation (CV) for hyperparameter tuning:

- 5 fold cv
- Using scikit-learn's grid search method
- Evaluation metric: Area under precision recall curve

Feature Selection: can be used using Lasso and then plotted for visualization

These steps can be piped together using sklearn pipeline class

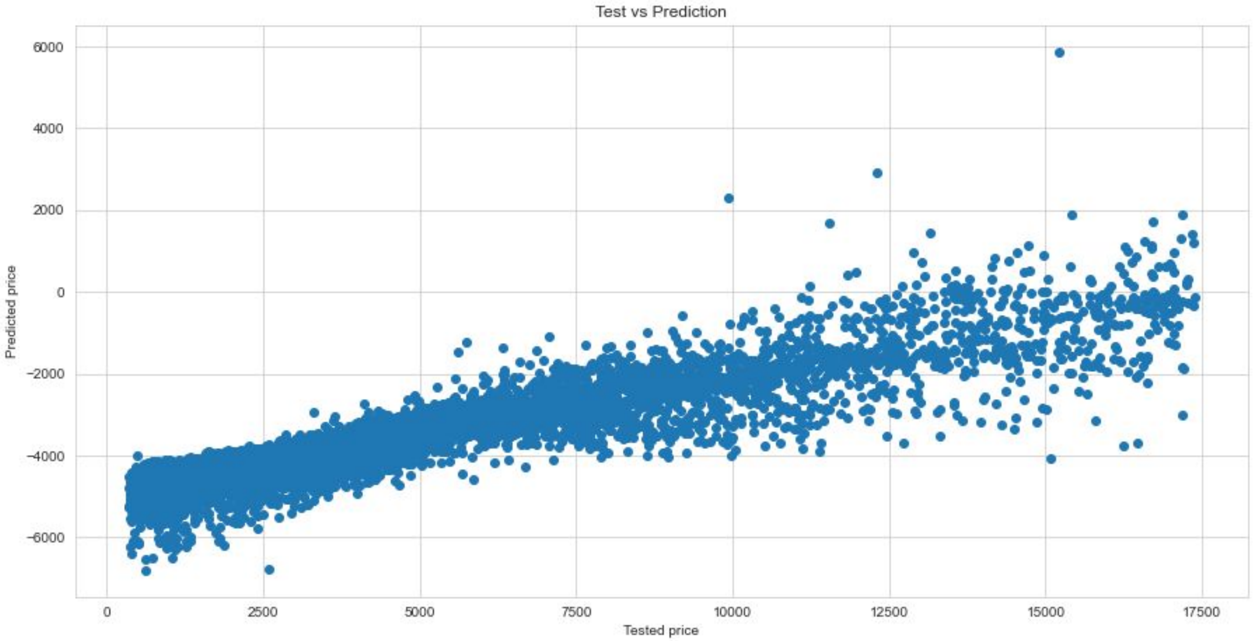
Testing using the same pipe (excluding cross validation)

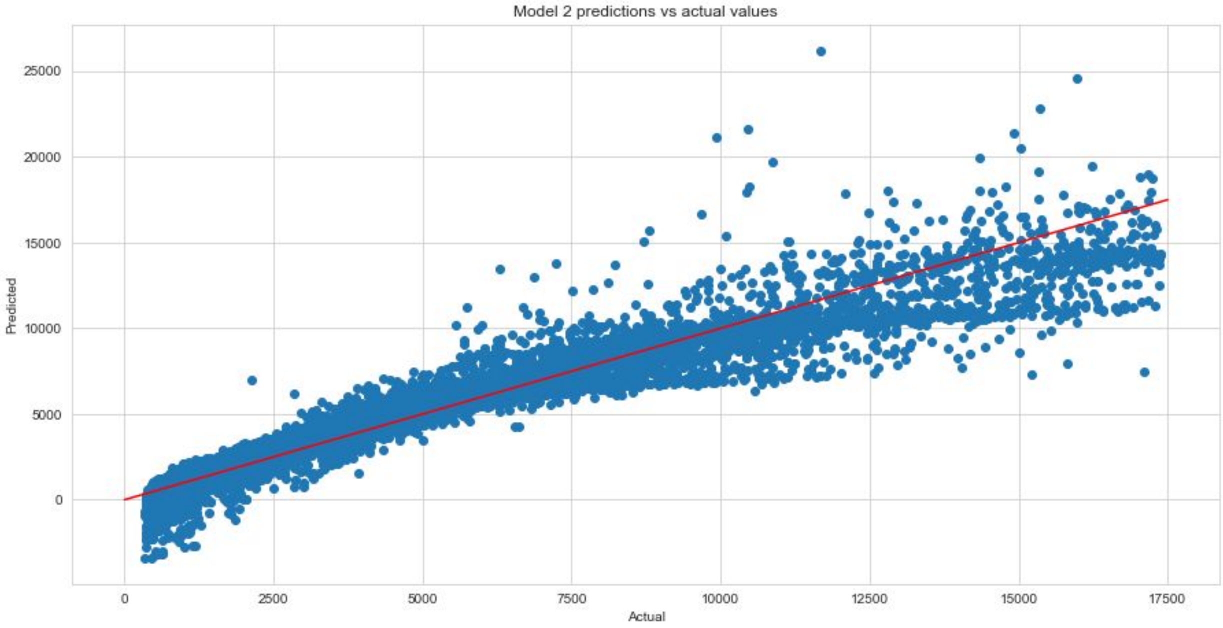
Performance evaluation using holdout dataset (20% of the whole data)

Regression Algorithms Used (Four Models):

1. Linear Regression – using sklearn
2. Linear Regression (Ordinary Least Squares) – using statsmodels
3. Ridge Regression – using sklearn
4. Lasso Regression – using sklearn

Model Comparisons





Logistic Regression using sklearn and Logistic Regression using Ordinary Least Squares

Details on the Best Model (ET)

Making a Ridge Regression model: Third model

Sklearn has a Ridge() function built into the linear_model module.

```
from sklearn.linear_model import Ridge

# Split data if not split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state=42)

# Initialize ridge model, normalize=True parameter assures all variables are scaled.
ridge = Ridge(alpha=0.1, normalize=True)

# We can use the scaled data from above to fit
ridge.fit(X_train, y_train)

# Make predictions
ridge_pred= ridge.predict(X_test)

# Score predictions
ridge.score(X_test, y_test)
```

```
0.8817195230671278
```

- By using the Ridge Regression Model we were able to increase the predictions of the predictive power to about 88%.

Making a Lasso Regression model: Fourth model

Sklearn has a Lasso() function built into the linear_model module.

```
from sklearn.linear_model import Lasso

# Split data if not split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state=42)

# Initialize ridge model, normalize=True parameter assures all variables are scaled.
lasso = Lasso(alpha=0.1, normalize=True)

# We can use the scaled data from above to fit
lasso.fit(X_train, y_train)

# Make predictions
lasso_pred= lasso.predict(X_test)

# Score predictions
lasso.score(X_test, y_test)
```

```
0.909819917007543
```

- By using the Lasso Regression Model we were able to increase the predictions of the predictive power to about 91%.

While the first model scored the highest the fourth model is the best to used for added complexity.

Assumptions, Limitations and Disclaimers

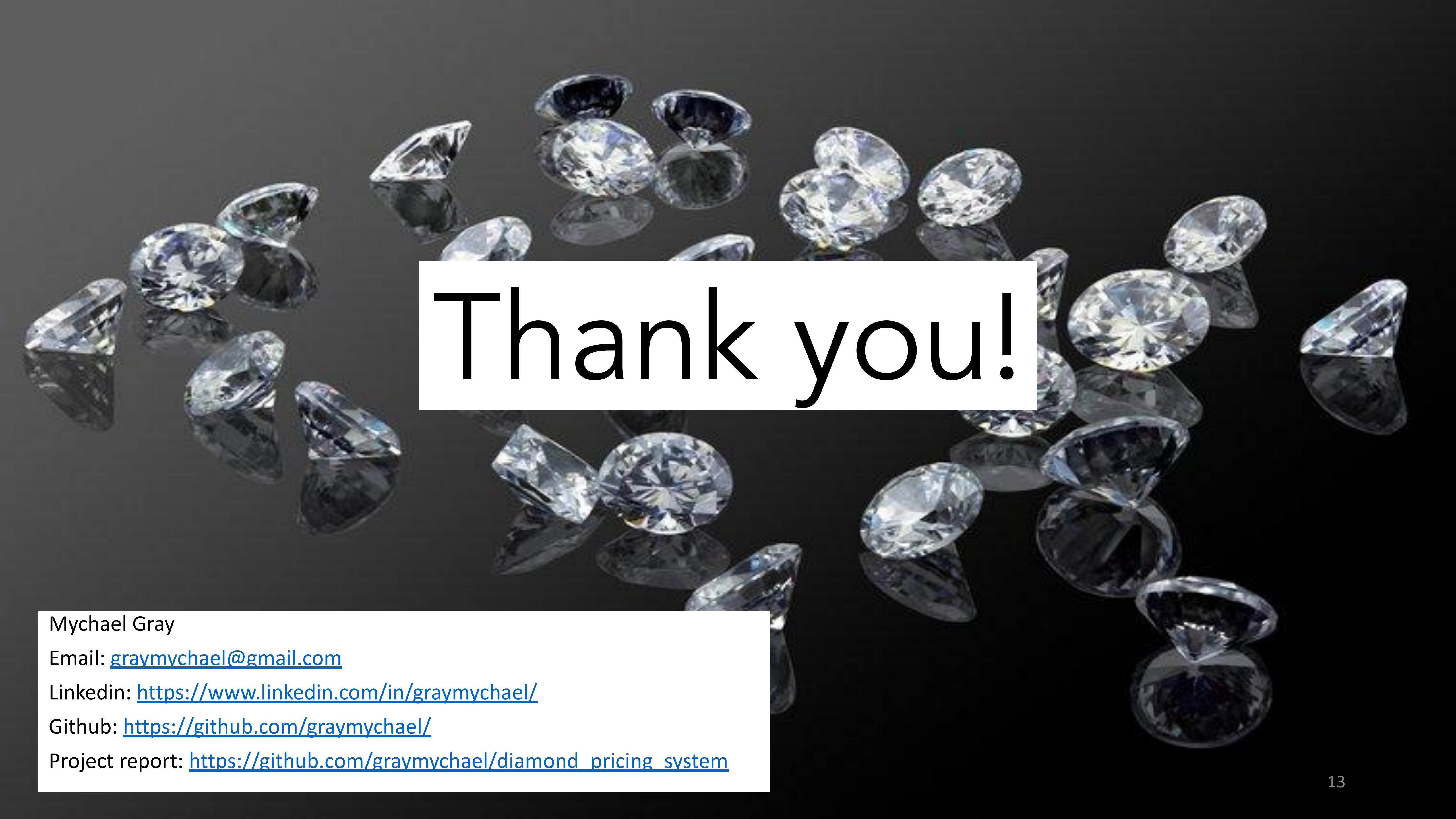
- We assume that all observations are independent, are assumed to be unrelated diamonds
- Used only 53940 observations of diamonds and past data that is not current
- The model will behave poorly if we try to predict prices of diamonds that are scheduled too far in future
- The diamonds data does not contain all data (several sources can be included in the future)

More Ideas to Improve the Model in Future

- Engineer more features related with diamonds such as customer feedback, customer surveys, environmental/social impact (blood diamonds), etc.
- Answer which diamonds are more volatile in price or more stable and why?
- Use social network and news media to extract sentiments about diamonds and stock market/economy health to get more features

Conclusions

- All sources of datasets contributed to the predictive power of the model.
- Out of 5 supervised classification models, the Extremely Randomized Trees provided the best results.
- Out of 67 features, we used only 31 features for the best model with 12 from the flight data, 16 from the weather data and 3 from the flight historical performances data (which we engineered).
- With 50%-50% splitting, the test data set gave ROC AUC = 0.89.



Thank you!

Mychael Gray

Email: graymychael@gmail.com

Linkedin: https://www.linkedin.com/in/graymychael/

Github: https://github.com/graymychael/

Project report: https://github.com/graymychael/diamond_pricing_system