Predicting Wildfires

May 17, 2025

0.0.1 Forest Fire Background Info

Forest fires have long been a major concern, especially in recent years in the drier regions of the United States. The impacts of these fires can be quite devastating because they can be very difficult to predict. Therefore, in this project, we will be creating 4 different types of prediction models (a tool that makes predictions from data): linear model, random forest model, decision tree model, and support vector machine model. We will compare the accuracy of each model to see which model is the best to use when predicting forest fires. We will also be using meteorological data from the data set below to create and test those models. The goal of this project is to create an accurate prediction model that can also be improved upon by others who are interested in predicting forest fires.

[12]:
```
import pandas as pd

data = pd.read_csv('forestfires.csv') # This will read in the data set.

data.head() # This shows us a sample of the head of the data set.
```

[12]:

	X	Y	month	day	FFMC	DMC	DC	ISI	temp	RH	wind	rain	area
0	7	5	mar	fri	86.2	26.2	94.3	5.1	8.2	51	6.7	0.0	0.0
1	7	4	oct	tue	90.6	35.4	669.1	6.7	18.0	33	0.9	0.0	0.0
2	7	4	oct	sat	90.6	43.7	686.9	6.7	14.6	33	1.3	0.0	0.0
3	8	6	mar	fri	91.7	33.3	77.5	9.0	8.3	97	4.0	0.2	0.0
4	8	6	mar	sun	89.3	51.3	102.2	9.6	11.4	99	1.8	0.0	0.0

0.0.2 Explaining the Data Set/Methods Used

The meteorological data above was collected from Montesinho Natural Park, Portugal (2000–2003). The data set above was publicly posted on Kaggle by Paolo Cortez and Anbal Morais. The independent variables that we will be focusing on for our model building are: temp (measured in Celsius), RH (Relative humidity measured as a percentage), wind (measured in km/h), and rain (measured in mm/mm^2). The dependent variable that I will be using is area (measured in hectares). I will be using 4 types of models: linear regression, random forest regression, decision tree regression, and support vector machines. The third-party libraries that I will be using are: numpy, pandas, matplotlib, seaborn, and scikit-learn. Lastly, I will be using a random state of 42 so that the results are reproducible when you run the code. Random state is a parameter in scikit-learn that controls the randomness of functions that involve randomization. I chose 42 because it is the most common seed (initial value used to initialize a random number generator) that many developers use when setting a seed.

0.0.3 Preprocessing

The only preprocessing step involved was applying a logarthimic transfomration on the area variable. A logarithmic transformation converts larger numbers into smaller values. This was necessary for this project because the original area distribution on the left graph shows us that the area variable is heavily skewed. This means that the area variance is unstable. The right figure shows a more stable variance of the areas after applying logarithmic transformation. This was necessary for our 4 regression models because not doing so will lead to the underprediction of large fires and the overprediction of smaller fires.

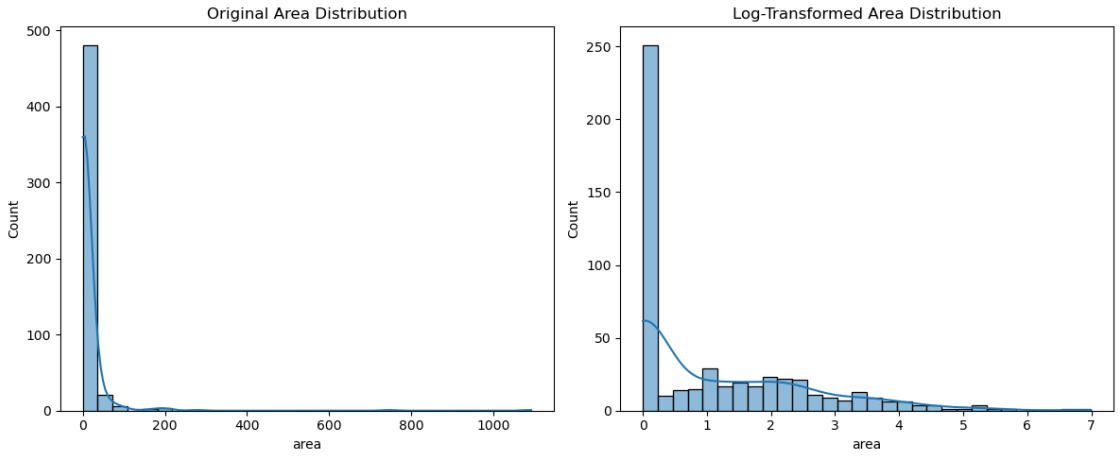
[7]:
```
features = ['temp', 'RH', 'wind', 'rain'] # Choosing our independent variables.
X = data[features] # This will subset
y_raw = data['area'] # This will subset our
y_log = np.log1p(y_raw)  # log(1 + area)


# This code chunk visualizes the original area distribution and the␣
 ↪log-transformed area distribution
plt.figure(figsize=(12, 5))

plt.subplot(1, 2, 1)
sns.histplot(y_raw, bins=30, kde=True)
plt.title('Original Area Distribution')

plt.subplot(1, 2, 2)
sns.histplot(y_log, bins=30, kde=True)
plt.title('Log-Transformed Area Distribution')

plt.tight_layout()
plt.show()
```



0.0.4 How we will be evaluating our 4 models

0.0.5 Statistical Metrics

We will be evaluating the models based on 4 numerics: mean RMSE (log), std RMSE (log), mean R2 (log), and std R2 (log). We will also be comparing the models before and after applying logarithmic transformation.

Mean RMSE (average root mean squared across k-folds) is a metric that tells you the size of the prediction error of your model. This is important because it shows you the errors of your prediction model. Rule of thumb: the lower the better.

Std RMSE (Standard deviation of RMSE across k-folds) is a metric that tells you how stable your model is across different subsets of data. This is important because it shows you the consistency of your model's prediction error. Rule of thumb: the lower the better.

Mean R2 (average R2 score across k-folds) is a metric that tells you how much variance your model explains. This is important because it tells you how well your model predicts new data. We will mostly be focusing on the R2 value when evaluating the model's overall performance. Rule of thumb: the closer to one the better.

Std R2 (Standard deviation of R2 across k-folds) is a metric that tells you how much your model's explanatory power varies across different train/test splits. This is important becomes it shows you how consistent your model is when it predicts new data. Rule of thumb: the lower the better.

0.0.6 Methodology

We will first use a single train/test split when building and testing each of our models. Train/test split is a technique that splits your data set into two parts: training set and testing set. The training set is used for building the model. The testing set is used to test your model with unseen data. This basic technique was used because it allows us to quickly build and test our models. 20 percent of the data will be used as testing data, and 80 percent of the data will be used as training data. I chose these We will just be looking at the RMSE and R2 scores for each model. We will not look at the standard deviation of those two metrics because there is no variability to be measured for a single train/test split.

We will then use k-fold cross validation when building and testing each of our models. K-fold cross-validation is a type of technique that splits the data set into K equal folds. The k-1 folds trains the model, while the final fold is used to test the model. We will use this technique because it will give us a more accurate representation of our models rather than using a single train/test split. K-fold validation gives us more accurate results because it splits the data into multiple sets rather than a single train/test split. We will be using 5 folds for k-fold. I used 5 folds because it's not too small to the point it creates inconsistent results, and it doesn't slow down the overall process too much.

We will also be hypertuning our models besides the linear regression model, which has no hyper-parameters. Hyperparameter tuning (hypertuning) is the process of searching for the best set of hyperparameters that lead to the most accurate or optimal performance of a machine learning model. Hypertuning is necessary because it will allow our models (excluding linear regression) to choose the best settings for optimal performance. Hypertuning also reduces the risk of overfitting and underfitting, which leads to better model accuracy and generalization.

We will be using GridSearchCV to find the best hypeparameters to use for our models (excluding linear regression). GridSearchCV is a tool in scikit-learn that allows us to systematically test combinations of hyperparameters to find the best settings machine learning models such as random forest. GridSearchCV is necessary because it will allow us to quickly find the best hypeparameters to use for all of the models (excluding linear regression).

0.0.7 Analysis of our 4 models

0.0.8 Linear regression model

Linear regression is a method that predicts the value of a dependent variable based on the linear relationship between one or more variables. It then uses those relationships to fit a linear equation to observed data. The advantages of using a linear regression model are: it's easy to understand, it's fast to train, and it works well the relationship is linear. The disadvantages of using linear regression are: it can't capture complex patterns, it doesn't work well for outliers, and it always assumes a linear relationship. The first step will be examining the model based on a single train/test split. Table 1 shows us the RMSE and R2 scores for both the training data and the testing data for each model. Table 1 also compares those scores before and after applying logarithmic transformation on the area variable. Table 1 shows us that the train RMSE is 45.337, the train R2 is 0.012, the test RMSE is 108.391, and the test R2 is 0.003 for the raw area variable. Table 1 also shows us the train RMSE is 46.484, the train R2 is -0.039, the test RMSE is 109.969, and the test R2 is -0.026 for the log-transformed area variable. This shows us that the linear regression model performed better on the training set than the testing set for the raw area variable due to the lower RMSE and the higher R2 score. This is expected because there was more data to work with for the training set. However, the linear regression model performed better on the testing set than the training set for the log-transformed area variable due to the higher R2 score. It also seems that applying a logarithmic transformation on the area variable leads to worse performing models. It also seems that applying a logarithmic transformation greatly improves the RMSE score for both the training and testing sets. This is because applying a logarithmic transformation reduces the impact of outliers and stabilizes the variance in the data. The next step is to observe the model based on a k-fold cross validation of 5 folds. K-fold cross validation will give us a more accurate representation of the model instead of relying on a single train/test split. Tables 2 and 3 will show us the mean RMSE, std RMSE, mean R2, and std R2 values for each model before and after applying logarithmic transformation on the area variable. Table 2 shows us that the mean RMSE is 40.88, the std RMSE is 21.79, the mean R2 is -0.18, and the std R2 is 0.32 for raw area variable. Table 3 shows us that the mean RMSE is 1.54, the std RMSE is 0.24, the mean R2 is -0.44, and the std R2 is 0.85 for log-transformed area variable. These two tables show us that the linear regression model performed better for the raw area variable than the log-transformed area variable overall. This is because the mean R2 and std R2 showed slighly worse scores when using the log-transformed area variable. However, the mean RMSE and std RMSE showed much better scores when using the log-transformed area variable as the dependent variable. This is also because applying a logarithmic transformation reduces the impact of outliers and stabilizes the variance in the data. Overall, linear regression is a very poor model to use regardless of using k-fold or using a single train/test split due to the poor R2 values.

0.0.9 Decision tree model

Decision tree regression is a type of regression that predicts continuous numerical values by creating a tree-like structure. Specifically, it splits data into subsets based on feature values, creating a tree-like structure where each internal node represents a decision on a feature, each branch represents an

outcome, and each leaf node holds a prediction. The advantages of using decision tree regression are: it's easy to visualize, it captures nonlinear patterns, and it handles both numerical and categorical data. The disadvantages of using decision tree regression are: it can over fit the data, it can be unstable, and it is usually not as accurate as other models. The first step will be examining the model based on a single train/test split. Table 1 shows us that the train RMSE is 42.40, the train R2 is 0.14, the test RMSE is 111.76, and the test R2 is -0.06 for the raw area variable. Table 1 also shows us the train RMSE is 46.135, the train R2 is -0.02, the test RMSE is 109.998, and the test R2 is -0.026 for the log-transformed area variable. These scores show us that the decision tree model performed better on the training set than the testing set for the raw area variable due to the higher R2 score for the training set. These scores also tell us that the decision tree model performed better on the training set than the testing set for the log-transformed area variable due to the slighly higher R2 score for the training set. The next step is to use to observe the model based on a k-fold cross validation of 5 folds. We will be also comparing the results between the untuned models and the tuned models. The hyperparameters that will be used for the raw area variable is: max_depth: 3, min_samples_split: 10. The hyperparameters that will be used for the log-transformed area variable is: max_depth: 3, min_samples_split: 5. These hyperparameters were found using GridSearchCV, which allow us to quickly find the best hyperparameters to use for our model. Tables 2 and 3 shows us the results for the untuned models. Tables 4 and 5 show us the results for the tuned models. Table 2 shows us that the mean RMSE is 63.2, the std RMSE is 20.43, the mean R2 is -4.75, and the std R2 is 6.89 for raw area variable. Table 3 shows us that the mean RMSE is 1.54, the std RMSE is 0.24, the mean R2 is -0.44, and the std R2 is 0.85 for log-transformed area variable. Table 4 shows us that the mean RMSE is 45.46, the std RMSE is 20.02, the mean R2 is -0.85, and the std R2 is 1.38 for raw area variable. Table 5 shows us that the mean RMSE is 1.48, the std RMSE is 0.04, the mean R2 is -0.19, and the std R2 is 0.19 for log-transformed area variable. These values tell us that the decision tree model doesn't perform well for unseen data due to the poor R2 scores for both the raw area variable and the log-transformed area variable. These values also tell us that tuning the model with the hyperparameters found via GridSearchCV slightly increases the R2 scores. Overall, the decision tree model performed poorly regardless of using k-fold or using a single train/test split due to the poor R2 values.

0.0.10 Random Forest model

Random forest regression is a type of regression that builds multiple decision trees and combines their outputs to improve accuracy and robustness. Specifically, it averages predictions from prediction trees. The advantages of using random forest regression are: it is very accurate, it handles non-linear data, and it handles outliers very well. The disadvantages of using random forest regression are: it's harder to interpret and it is slow to train and predict. The first step will be examining the model based on a single train/test split. Table 1 shows us that the train RMSE is 32.914, the train R2 is 0.479, the test RMSE is 109.304, and the test R2 is -0.014 for the raw area variable. Table 1 also shows us the train RMSE is 46.282, the train R2 is -0.030, the test RMSE is 109.938, and the test R2 is -0.025 for the log-transformed area variable. These scores show us that the random forest model performed better on the training set than the testing set for the raw area variable due to the higher R2 score for the training set. These scores also tell us that the random forest model performed better on the testing set than the training set for the log-transformed area variable due to the slighly higher R2 score for the testing set. The next step is to use to observe the model based on a k-fold cross validation of 5 folds. We will be also comparing the results between the untuned models and the tuned models. The hyperparameters that will be used for the raw area variable is: max_depth: 3, n_estimators: 200. The hyperparameters that will be used for the

log-transformed area variable is: max_depth: 3, n_estimators: 50. These hyperparameters were found using GridSearchCV, which allow us to quickly find the best hyperparameters to use for our model. Tables 2 and 3 shows us the results for the untuned models. Tables 4 and 5 show us the results for the tuned models. Table 2 shows us that the mean RMSE is 47.06, the std RMSE is 19.95, the mean R2 is -0.90, and the std R2 is 1.20 for raw area variable. Table 3 shows us that the mean RMSE is 1.51, the std RMSE is 0.10, the mean R2 is -0.23, and the std R2 is 0.07 for log-transformed area variable. Table 4 shows us that the mean RMSE is 43.75, the std RMSE is 20.91, the mean R2 is -0.45, and the std R2 is 0.55 for raw area variable. Table 5 shows us that the mean RMSE is 1.38, the std RMSE is 0.13, the mean R2 is -0.02, and the std R2 is 0.03 for log-transformed area variable. These values tell us that the random forest model doesn't perform well for unseen data due to the poor R2 scores for both the raw area variable and the log-transformed area variable. These values also tell us that tuning the model with the hyperparameters found via GridSearchCV slightly increases the R2 scores. Overall, the random forest model performed poorly regardless of using k-fold or using a single train/test split due to the poor R2 values.

0.0.11 Support Vector Machines

Support vector machines is a type of regression that finds the optimal boundary that best separates the data into classes. Specifcally, it fits a line within a margin of tolerance around the actual values. The advantages of using support vector machines are: it is captures high-dimensional data really well, can capture very complex relationships, and it works even when the data isn't completely separated. The disadvantages of using support vector machines are: it's very slow on large data sets and it is very hard to explain. Table 1 shows us that the train RMSE is 46.670, the train R2 is -0.047, the test RMSE is 110.167, and the test R2 is -0.030 for the raw area variable. Table 1 also shows us the train RMSE is 46.692, the train R2 is -0.048, the test RMSE is 110.174, and the test R2 is -0.030 for the log-transformed area variable. These scores show us that the support vector machines model performed better on the testing set than the training set for the raw area variable due to the higher R2 score for the training set. These scores also tell us that the support vector machines model performed better on the testing set than the training set for the log-transformed area variable due to the slighly higher R2 score for the testing set. The next step is to use to observe the model based on a k-fold cross validation of 5 folds. We will be also comparing the results between the untuned models and the tuned models. The hyperparameters that will be used for the raw area variable is: C: 100, epsilon: 0.5, and kernel: rbf. The hyperparameters that will be used for the log-transformed area variable is: C: 100, epsilon: 0.5, and kernel: rbf. These hyperparameters were found using GridSearchCV, which allow us to quickly find the best hyperparameters to use for our model. Tables 2 and 3 shows us the results for the untuned models. Tables 4 and 5 show us the results for the tuned models. Table 2 shows us that the mean RMSE is 40.85, the std RMSE is 23.08, the mean R2 is -0.10, and the std R2 is 0.06 for raw area variable. Table 3 shows us that the mean RMSE is 1.52, the std RMSE is 0.16, the mean R2 is -0.24, and the std R2 is 0.09 for log-transformed area variable. Table 4 shows us that the mean RMSE is 40.70, the std RMSE is 23.12, the mean R2 is -0.09, and the std R2 is 0.05 for raw area variable. Table 5 shows us that the mean RMSE is 1.43, the std RMSE is 0.17, the mean R2 is -0.10, and the std R2 is 0.09 for log-transformed area variable. These values tell us that the support vector machines model doesn't perform well for unseen data due to the poor R2 scores for both the raw area variable and the log-transformed area variable. These values also tell us that tuning the model with the hyperparameters found via GridSearchCV slightly increases the R2 scores. Overall, the support vector machines model performed poorly regardless of using k-fold or using a single train/test split due to the poor R2 values.

0.0.12 Table 1

[5]:
```
with open('/home/jovyan/great/FinalProject/results/train_test_scores.txt', 'r')
as f:
    content = f.read()
    print(content)
```

```
--- Training vs Testing Scores (RAW) ---
--- Decision Tree ---
Train RMSE: 42.398, Test RMSE: 111.757
Train R²: 0.136, Test R²: -0.060

--- Random Forest ---
Train RMSE: 32.914, Test RMSE: 109.304
Train R²: 0.479, Test R²: -0.014

--- SVR ---
Train RMSE: 46.670, Test RMSE: 110.167
Train R²: -0.047, Test R²: -0.030

--- Linear Regression ---
Train RMSE: 45.337, Test RMSE: 108.391
Train R²: 0.012, Test R²: 0.003

--- Training vs Testing Scores (LOG-TRANSFORMED) ---
--- Decision Tree ---
Train RMSE: 46.135, Test RMSE: 109.998
Train R²: -0.023, Test R²: -0.026

--- Random Forest ---
Train RMSE: 46.282, Test RMSE: 109.938
Train R²: -0.030, Test R²: -0.025

--- SVR ---
Train RMSE: 46.692, Test RMSE: 110.174
Train R²: -0.048, Test R²: -0.030

--- Linear Regression ---
Train RMSE: 46.484, Test RMSE: 109.969
Train R²: -0.039, Test R²: -0.026
```

0.0.13 Table 2

[6]:
```
with open('/home/jovyan/great/FinalProject/results/untuned_raw.txt', 'r')
as f:
    content = f.read()
    print(content)
```

```
--- Untuned models on RAW target ---
                   Mean RMSE   Std RMSE   Mean R2    Std R2
Model
Linear Regression  40.884577  21.787281 -0.183589  0.324067
Decision Tree      63.200770  20.426981 -4.752024  6.888484
Random Forest      47.055108  19.954895 -0.899120  1.204614
SVR                40.846172  23.080156 -0.100132  0.062001
```

0.0.14 Table 3

[7]:
```
with open('/home/jovyan/great/FinalProject/results/untuned_log.txt', 'r')
as f:
    content = f.read()
    print(content)
```

```
--- Untuned models on LOG target ---
                   Mean RMSE  Std RMSE   Mean R2    Std R2
Model
Linear Regression   1.548826  0.241614 -0.438975  0.853782
Decision Tree       1.901854  0.113323 -0.966131  0.254800
Random Forest       1.510884  0.103124 -0.230445  0.071622
SVR                 1.523915  0.156390 -0.244750  0.094948
```

0.0.15 Table 4

[8]:
```
with open('/home/jovyan/great/FinalProject/results/tuned_raw.txt', 'r')
as f:
    content = f.read()
    print(content)
```

```
--- Tuned models on RAW target ---
Best params for Decision Tree: {'max_depth': 3, 'min_samples_split': 10}
Best params for Random Forest: {'max_depth': 3, 'n_estimators': 200}
Best params for SVR: {'C': 100, 'epsilon': 0.5, 'kernel': 'rbf'}
               Mean RMSE   Std RMSE   Mean R2    Std R2
Model
Decision Tree  45.462807  20.022810 -0.848815  1.383083
Random Forest  43.753043  20.907845 -0.445812  0.552955
SVR            40.699967  23.117058 -0.088286  0.054258
```

0.0.16 Table 5

[9]:
```
with open('/home/jovyan/great/FinalProject/results/tuned_log.txt', 'r')
as f:
    content = f.read()
    print(content)
```

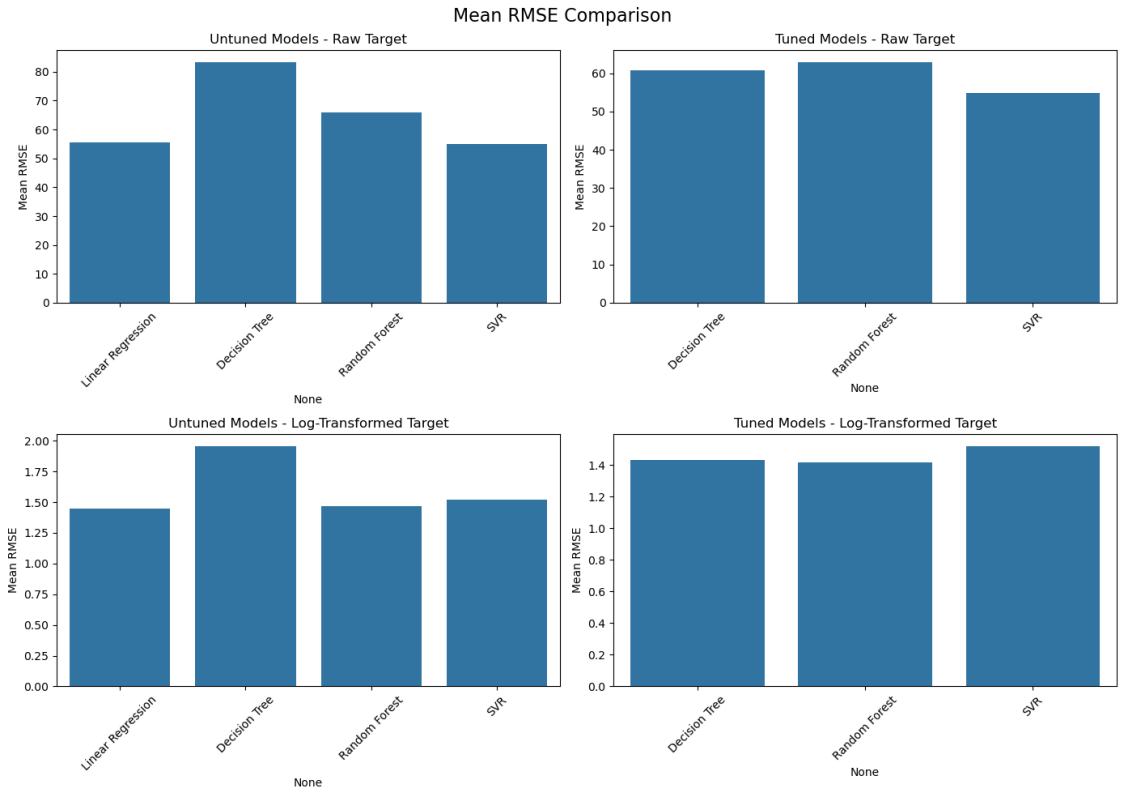
```
--- Tuned models on LOG target ---
Best params for Decision Tree: {'max_depth': 3, 'min_samples_split': 5}
Best params for Random Forest: {'max_depth': 3, 'n_estimators': 50}
Best params for SVR: {'C': 100, 'epsilon': 0.5, 'kernel': 'rbf'}
               Mean RMSE  Std RMSE   Mean R2    Std R2
Model
Decision Tree   1.475714  0.041947 -0.189496  0.185833
Random Forest   1.382318  0.126554 -0.024608  0.031966
SVR             1.431867  0.169200 -0.095107  0.087173
```

0.0.17 Conclusion/Demonstration

Figures 1–4 show the performance metrics (Mean RMSE, Mean R2, Std RMSE, and Std R2) for each regression model using a 5-fold cross-validation. The figures also compare the results for untuned and tuned models, and for both the raw area variable and log-transformed area variable. Figures 1-4 also demonstrate that none of the models performed adequately well due to the negative R2 scores for both the raw area variable and the log-transformed area variable. However, out of all the models, figure 3 shows us that the best performing model for the log-transformed area variable after hypertuning was the random forest model due to highest r2 score. Figure 3 also shows us that the best performing model for the raw area variable after hypertuning was the support vector machine model due to the highest r2 score. Hypetuning the models (excluding regression) seemed to improve each of the model's performance significantly due to the fact that optimal hyperparameters were used, which were found via GridSearchCV. Applying applying a logarthimic transfomration on the area variable seemed to improve the each model's R2 score due to the fact the logarithmic transformation stabilized the variance for the area variable. The worst perfoming model was decision tree regression due to the fact that it got the lowest R2 score for both the raw area variable and the log-transformed area variable according to table 3. For future analysis, I highly recommend that you include more independent variables for each of the models to test to see if that improves model performance. I would also increase your range of hypeparameters to test to see if that has any impact on the overall analysis. Overall, none the model performed particulary well due to the fact that the R2 values are always in the negatives.

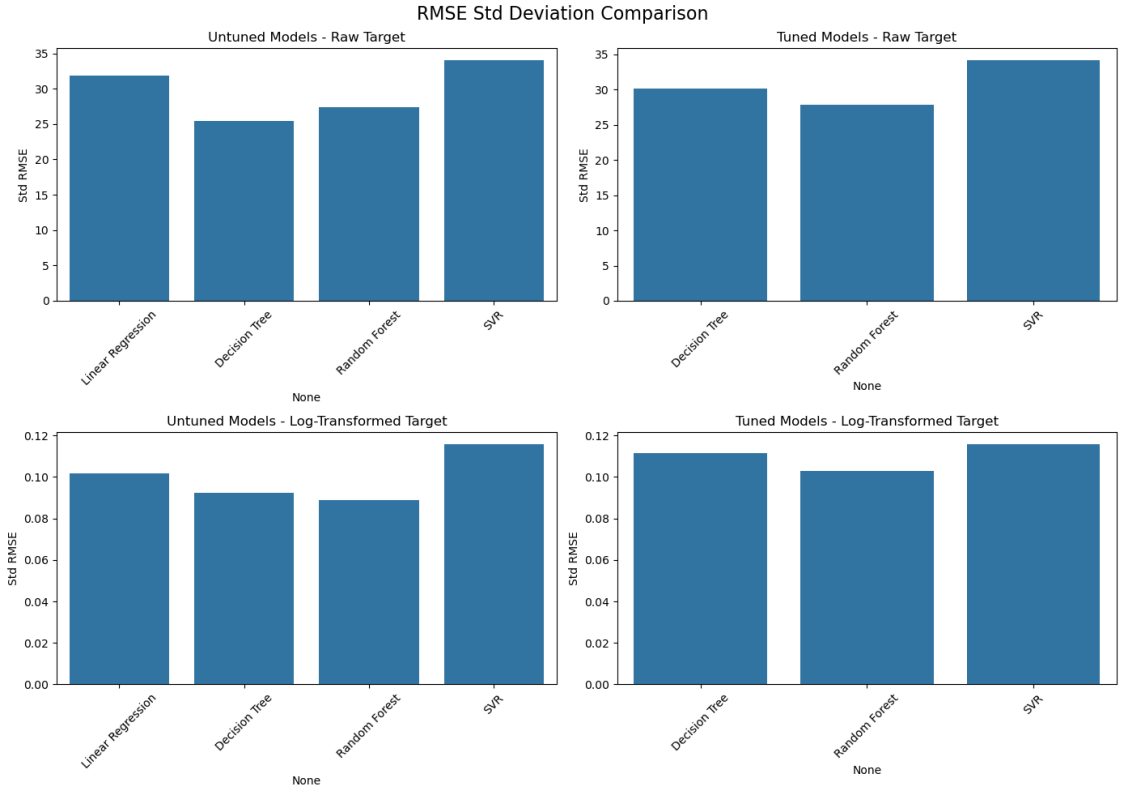
0.0.18 Figure 1

[30]:
```
plot_metric_grid(res_ur, res_tr, res_ul, res_tl, "Mean RMSE", "Mean RMSE",␣
↪"Mean RMSE Comparison")
```



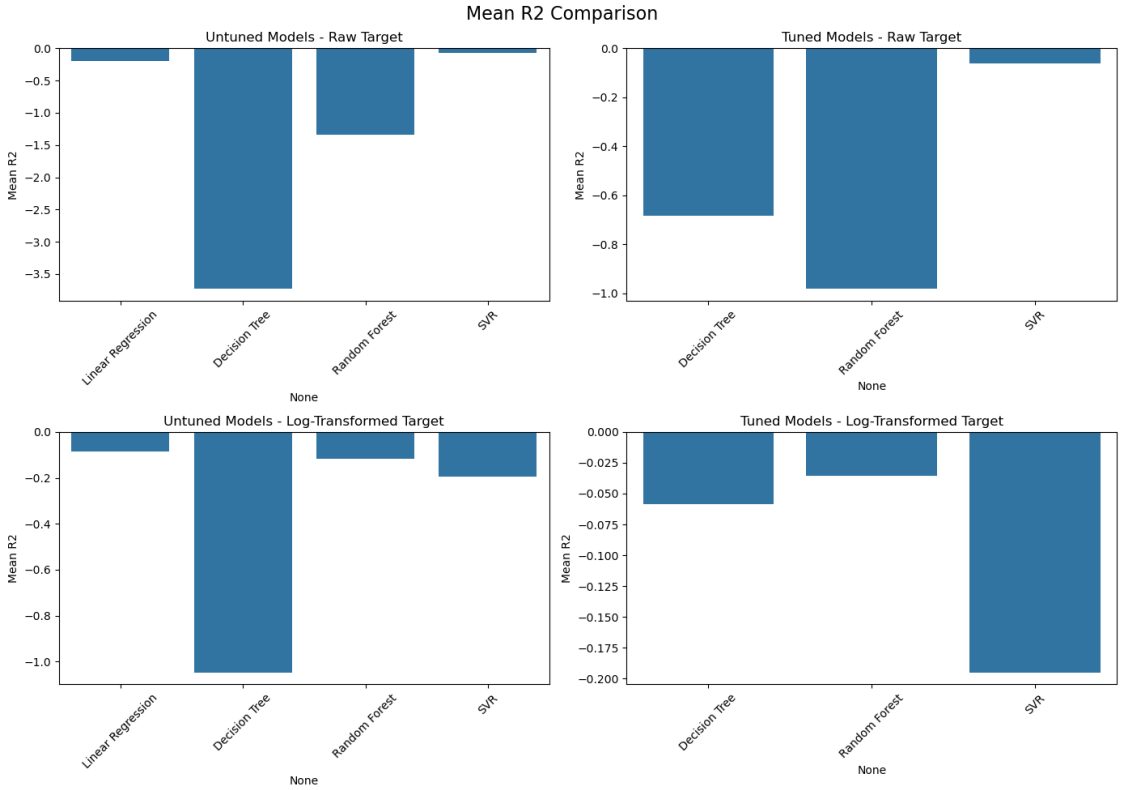
0.0.19 Figure 2

[31]:
```
plot_metric_grid(res_ur, res_tr, res_ul, res_tl, "Std RMSE", "Std RMSE", "RMSE␣
↪Std Deviation Comparison")
```



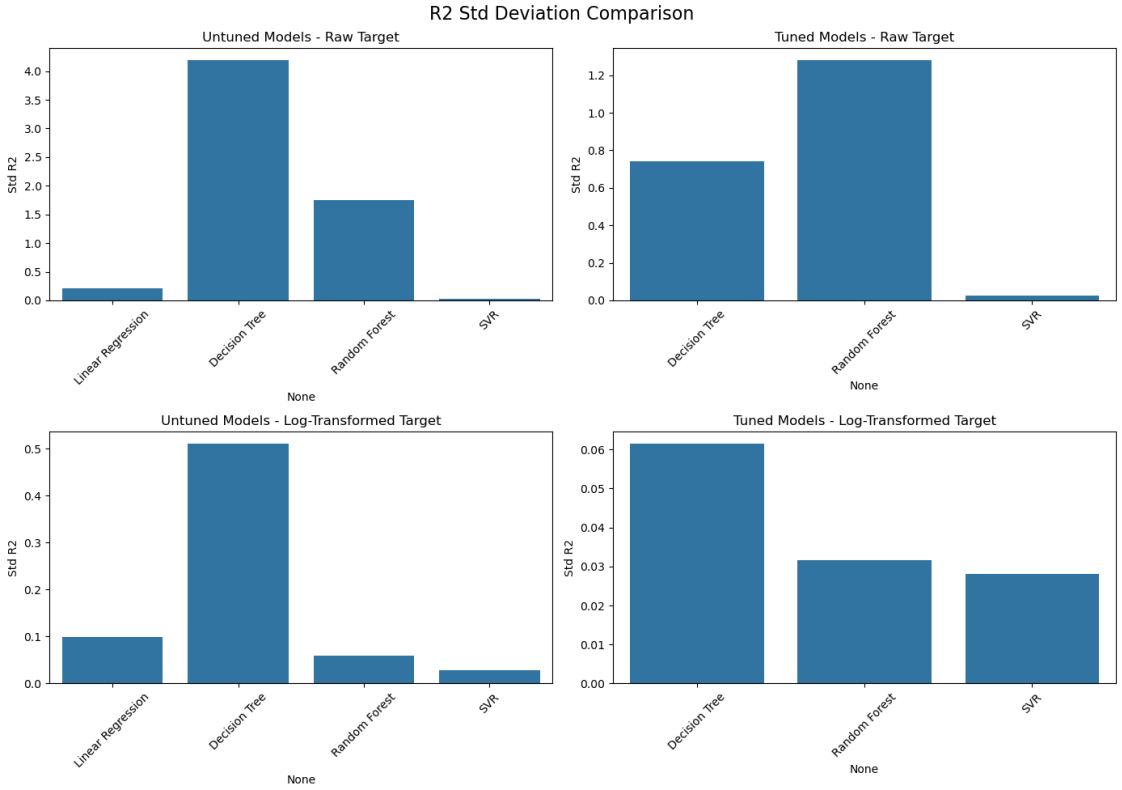
0.0.20 Figure 3

[32]:
```
plot_metric_grid(res_ur, res_tr, res_ul, res_tl, "Mean R2", "Mean R2", "Mean R2␣
↪Comparison")
```



0.0.21 Figure 4

[33]:
```
plot_metric_grid(res_ur, res_tr, res_ul, res_tl, "Std R2", "Std R2", "R2 Std␣
↪Deviation Comparison")
```



0.0.22 References

https://www.kaggle.com/datasets/elikplim/forest-fires-data-set