Using Correlation, Significance, and Variance Thresholds to select the best features to predict Soybean Yield

Bhalisa Sodo

Problem

In this presentation I attempt to find the variables that best influence soybean yield. There are more than 30 potential influencers/predictors of soybean yield in our dataset, and this number goes up to more than 70, when we dummy encode the 'DISTRICTS' variable.

The task is to use analytical feature selection methods to reduce the computational cost of predicting soybean yield, covariance and multicollinearity between variables, by narrowing the predictors to the most relevant or influential to soybean yield. Thereby curbing the 'curse of dimensionality' and getting quality results from ML models.

Data variables

- **Districts**: list of 46 districts of Madhya Pradesh State
- **Year**: list of years from 2010 to 2019
- **Yield**: soybean yield expressed in tonnes per hectare
- **NDVI**: average Normalized Difference Vegetation Index values extracted for 2010-2019
- **LAI**: average Leaf Area Index values extracted for 2010-2019
- **ET**: average Evapotranspiration values expressed in mm for 2010-2019
- **LST:** average Land Surface Temperature extracted for 2010-2019

Data cleaning & preprocessing

14	LAI_NOV	460	non-null	float64
15	ET_JUN	460	non-null	float64
16	ET_JUL	459	non-null	float64
17	ET_AUG	460	non-null	float64
18	ET_SEP	460	non-null	float64

By exploring the data using the info() function I discovered one null value in the 'ET_JUL' column.

14	LAI_NOV	460	non-null	float64
15	ET_JUN	460	non-null	float64
16	ET_JUL	460	non-null	float64
17	ET_AUG	460	non-null	float64
18	ET_SEP	460	non-null	float64

I remedied this by using the fillna(method='pad', inplace=True) function, which replaced the null value with a previous valid observation.

Data cleaning & preprocessing

```
df_dummies.columns
```

```
Index(['YEAR', 'NDVI_JUN', 'NDVI_JUL', 'NDVI_AUG', 'NDVI_SEP', 'NDVI_OCT',
       'NDVI_NOV', 'LAI_JUN', 'LAI_JUL', 'LAI_AUG', 'LAI_SEP', 'LAI_OCT',
       'LAI_NOV', 'ET_JUN', 'ET_JUL', 'ET_AUG', 'ET_SEP', 'ET_OCT', 'ET_NOV',
       'LST_JUN', 'LST_JUL', 'LST_AUG', 'LST_SEP', 'LST_OCT', 'LST_NOV',
       'RF_JUN', 'RF_JUL', 'RF_AUG', 'RF_SEP', 'RF_OCT', 'RF_NOV',
       'DISTRICTS_Ashoknagar', 'DISTRICTS_Balaghat', 'DISTRICTS_Barwani',
       'DISTRICTS_Betul', 'DISTRICTS_Bhind', 'DISTRICTS_Bhopal',
       'DISTRICTS_Burhanpur', 'DISTRICTS_Chhatarpur', 'DISTRICTS_Chhindwara',
       'DISTRICTS_Damoh', 'DISTRICTS_Datia', 'DISTRICTS_Dewas',
       'DISTRICTS_Dhar', 'DISTRICTS_Dindori', 'DISTRICTS_Guna',
       'DISTRICTS_Gwalior', 'DISTRICTS_Harda', 'DISTRICTS_Hoshangabad',
       'DISTRICTS_Indore', 'DISTRICTS_Jabalpur', 'DISTRICTS_Jhabua',
       'DISTRICTS_Katni', 'DISTRICTS_Mandla', 'DISTRICTS_Mandsaur',
       'DISTRICTS_Morena', 'DISTRICTS_Narsinghpur', 'DISTRICTS_Neemuch',
       'DISTRICTS_Panna', 'DISTRICTS_Raisen', 'DISTRICTS_Rajgarh',
       'DISTRICTS_Ratlam', 'DISTRICTS_Rewa', 'DISTRICTS_Sagar',
       'DISTRICTS_Satna', 'DISTRICTS_Sehore', 'DISTRICTS_Seoni',
       'DISTRICTS_Shahdol', 'DISTRICTS_Shajapur', 'DISTRICTS_Sheopur',
       'DISTRICTS_Shivpuri', 'DISTRICTS_Sidhi', 'DISTRICTS_Tikamgarh',
       'DISTRICTS_Ujjain', 'DISTRICTS_Umaria', 'DISTRICTS_Vidisha', 'YIELD'],
      dtype='object')
```

Since 'DISTRICTS' column is categorical, I had to encode it using *pd.get_dummies(df, drop_first=True)*. Which took us from a count of 33 columns to 77 columns. Where the resulting values would be either 0 or 1, in each new column's rows.

RandomForestRegressor before feature selection

```
X_scaled
```

```
array([[-1.5666989 , -0.34636311,  0.66073433, ..., -0.1490712 ,
        -0.1490712 , -0.1490712 ],
       [-1.5666989 , -1.32594862,  0.85974748, ..., -0.1490712 ,
        -0.1490712 , -0.1490712 ],
       [-1.5666989 ,  2.90596323,  0.11789509, ..., -0.1490712 ,
        -0.1490712 , -0.1490712 ],
       ...,
       [ 1.5666989 , -0.80082328,  0.90207201, ...,  6.70820393,
        -0.1490712 , -0.1490712 ],
       [ 1.5666989 ,  1.40560512,  1.86319545, ..., -0.1490712 ,
         6.70820393, -0.1490712 ],
       [ 1.5666989 , -1.0276998 ,  0.57801804, ..., -0.1490712 ,
        -0.1490712 ,  6.70820393]])
```

```
# Get predictions
y_pred = RF.predict(x_test)

# Compute RMSE
print("Random Forest RMSE:",np.sqrt(mean_squared_error(y_test,y_pred)))
```

```
Random Forest RMSE: 0.3847609349923953
```

```
# Compute RMSE
print("Random Forest R^2:", r2_score(y_test, y_pred))
```

```
Random Forest R^2: 0.15065199871412382
```

I scaled the data before feeding it into the model because the standard deviations and means between variables are too far apart. I used *StandardScaler()*

After training the RandomForestRegressor, I got a Root Mean Squared Error (RMSE) of 0.3847, which shows the model is relatively accurate. R-squared is 0.1506, which is low. RMSE should be low and R-squared should be high to indicate better model accuracy.

Linear Regression before feature selection

```
# Compute RMSE
print("Linear Model RMSE:",np.sqrt(mean_squared_error(y_test,y_pred_lm_1)))
```

```
Linear Model RMSE: 0.37587234135643915
```

```
print("Linear Model R^2:", r2_score(y_test, y_pred_lm_1))
```

```
Linear Model R^2: 0.18944131272621378
```

I used the same training and testing sets for the Linear Regression model, and as we can see the RMSE and R-squared values are relatively lower and higher respectively, than the RandomForestRegressor. This means the linear regression model performed slightly better than the random forest regression model, in this unfiltered data. Now let us do some feature selection!

Feature Selection using Correlation and Significance

Firstly, I sorted the columns according to their correlation coefficients (pearsonr) and p-values / significance to the response variable, Yield.

	Correlation_Coefficient	P_Value
RF_JUN	-0.2477990780454609	0.0
RF_AUG	-0.2164941107600098	3e-06
LST_OCT	0.21250472230307713	4e-06
LST_SEP	0.19191536679703364	3.4e-05
LAI_JUL	-0.18943152104252955	4.3e-05
ET_OCT	-0.18874953681939524	4.6e-05
NDVI_OCT	-0.18617740599532706	5.9e-05
LST_NOV	0.18340553627619874	7.6e-05
ET_NOV	-0.15923493411825154	0.000608
DISTRICTS_Umaria	-0.1532206548357708	0.000978
DISTRICTS_Chhindwara	0.15289436302301623	0.001003
RF_OCT	-0.1506283149584223	0.001194
YEAR	-0.1468880153543424	0.001583
NDVI_JUL	-0.14671264442344212	0.001604
NDVI_NOV	-0.1457640118446057	0.001721
DISTRICTS_Gwalior	0.14425567345463472	0.001924
LST_JUL	0.13709343897009196	0.003217
ET_JUL	-0.1327159651393936	0.004354
LST_JUN	-0.12938976669495159	0.005449

Feature Selection using Correlation and Significance

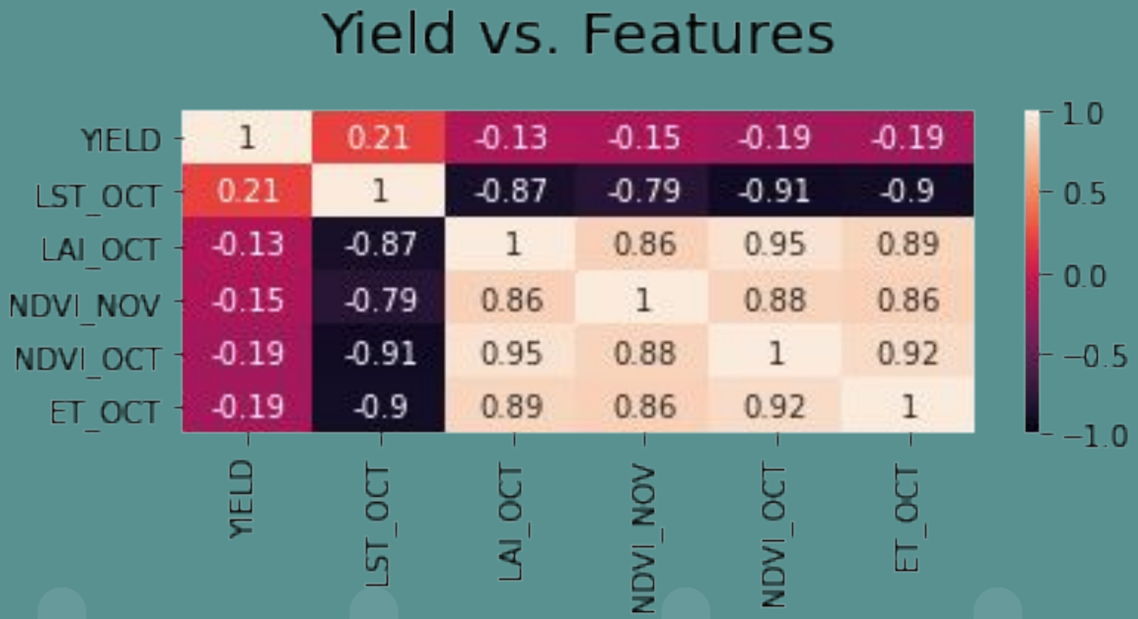
```
X_data.columns
```

```
Index(['LST_OCT', 'LST_SEP', 'LST_NOV', 'DISTRICTS_Chhindwara',
       'DISTRICTS_Gwalior', 'LST_JUL', 'DISTRICTS_Dhar', 'DISTRICTS_Morena',
       'NDVI_JUN', 'ET_AUG', 'DISTRICTS_Raisen', 'RF_SEP', 'DISTRICTS_Satna',
       'LAI_NOV', 'DISTRICTS_Sidhi', 'LAI_OCT', 'LST_JUN', 'ET_JUL',
       'NDVI_NOV', 'NDVI_JUL', 'YEAR', 'RF_OCT', 'DISTRICTS_Umaria', 'ET_NOV',
       'NDVI_OCT', 'ET_OCT', 'LAI_JUL', 'RF_AUG', 'RF_JUN'],
      dtype='object')
```

Previously, we obtained a sorted list of the p-values and correlation coefficients for each of the features, when considered on their own.

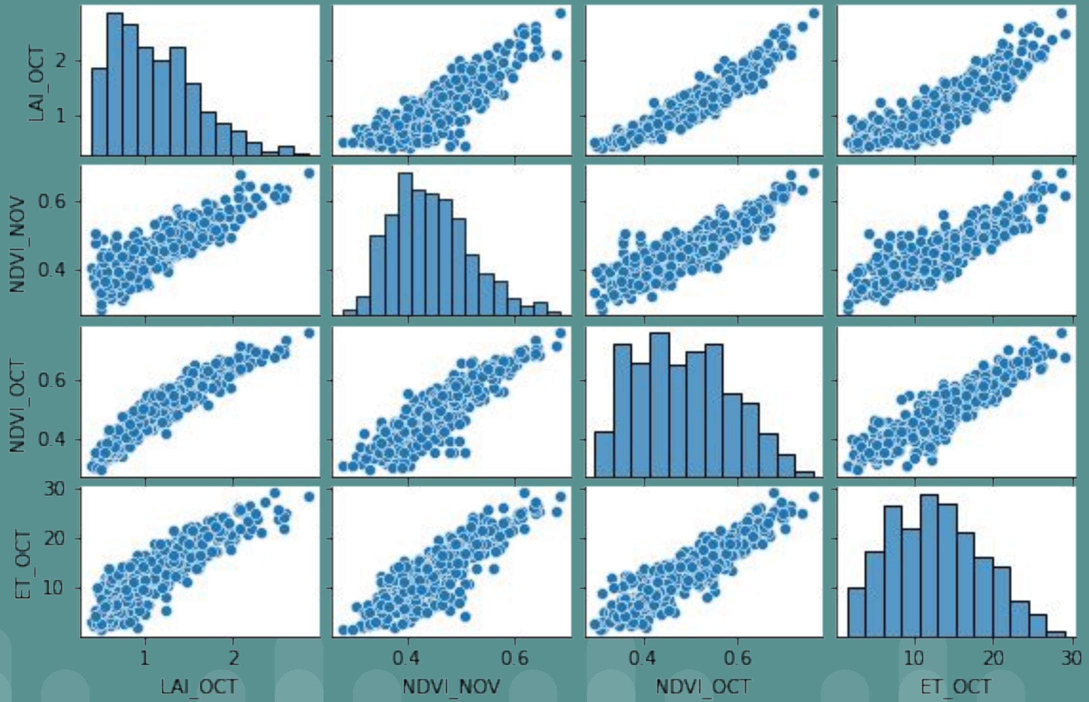
If we were to use a logic test with a significance value of 5% (p-value < 0.05), we could infer that the above features are the most statistically significant. 29 features out of 77.

Feature Selection using Correlation and Significance



The correlation heatmap shows a subset of the data, that contains features with a p-value < 0.05, and 0.09 < correlation < -0.09 between two or more features. This indicates multicollinearity amongst our features.

Feature Selection using Correlation and Significance



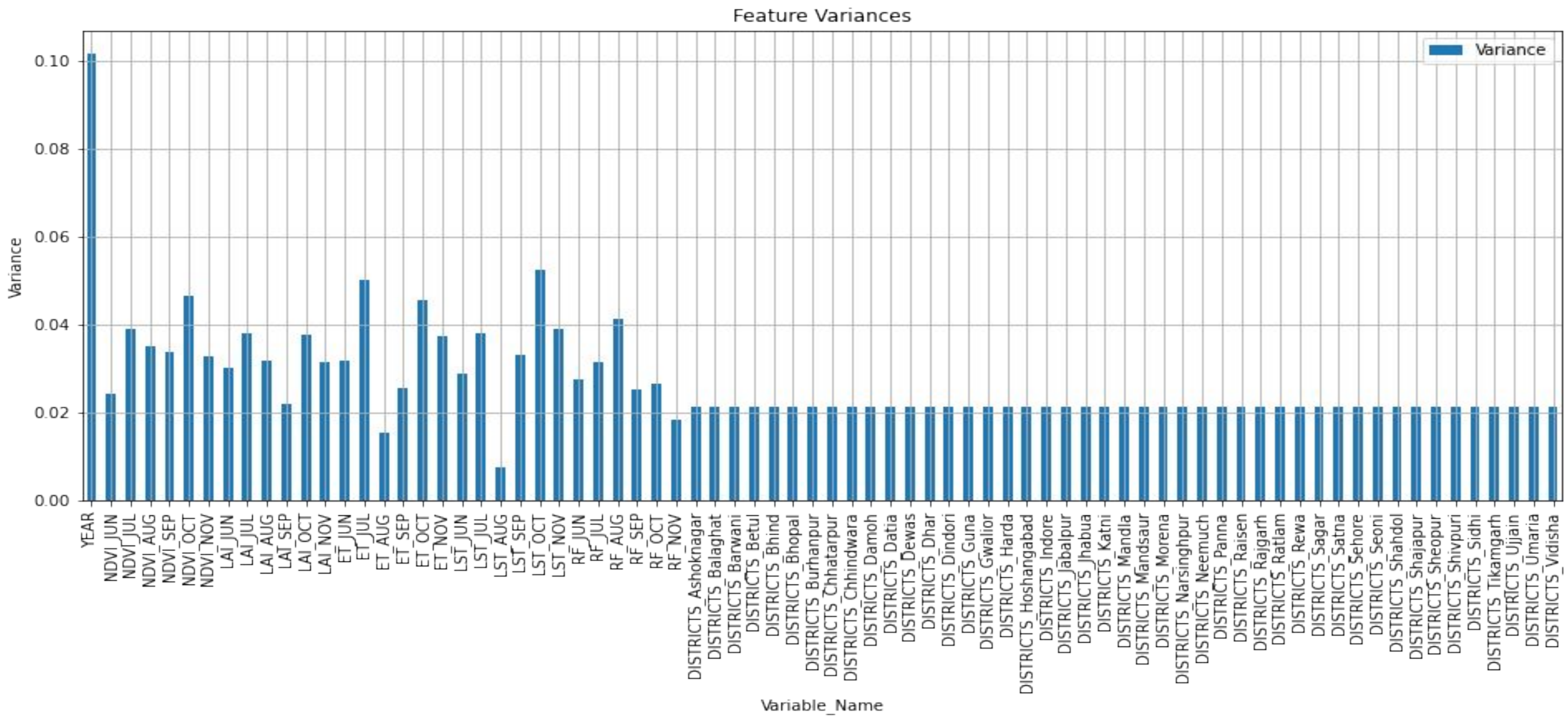
The pairplot shows the columns from the prev. Slide, a subset of the data, that contains features with a p-value < 0.05, and 0.09 < correlation < -0.09 between two or more features. This indicates multicollinearity amongst our features.

Feature Selection using Correlation and Significance

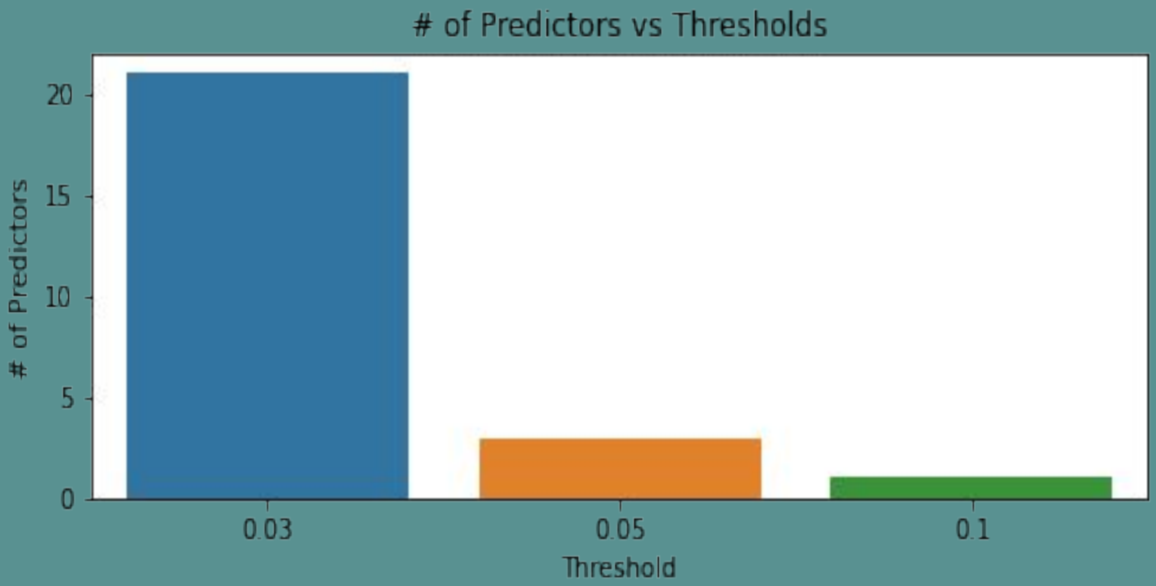
```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  YIELD   R-squared:                       0.328
Model:                            OLS   Adj. R-squared:                  0.290
Method:                 Least Squares   F-statistic:                     8.492
Date:                Tue, 04 Oct 2022   Prob (F-statistic):           6.83e-25
Time:                        09:34:09   Log-Likelihood:                -145.01
No. Observations:                 460   AIC:                             342.0
Df Residuals:                     434   BIC:                             449.4
Df Model:                          25
Covariance Type:            nonrobust
==============================================================================================
                        coef    std err          t      P>|t|      [0.025      0.975]
----------------------------------------------------------------------------------------------
Intercept            67.2043     17.361      3.871      0.000      33.081     101.327
```

After having filtered the data that contains features with a p-value < 0.05, and 0.09 < correlation < -0.09 between two or more features. **I observe a 0.328 R-squared, from Ordinary Least Squares, a quite significant jump compared to the RandomForestRegressor and Linear Regression models.**

Feature Selection using Variance Threshold

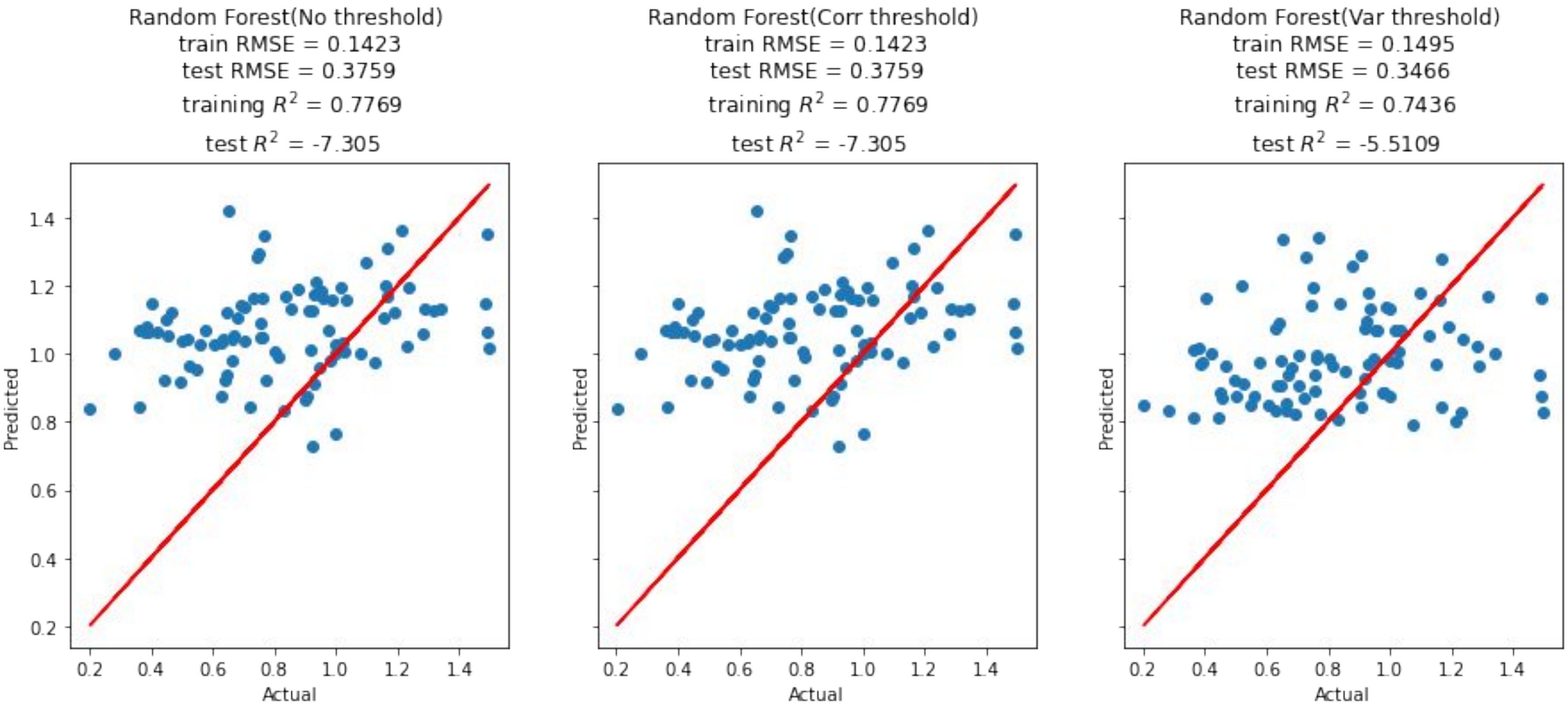


Feature Selection using Variance Threshold

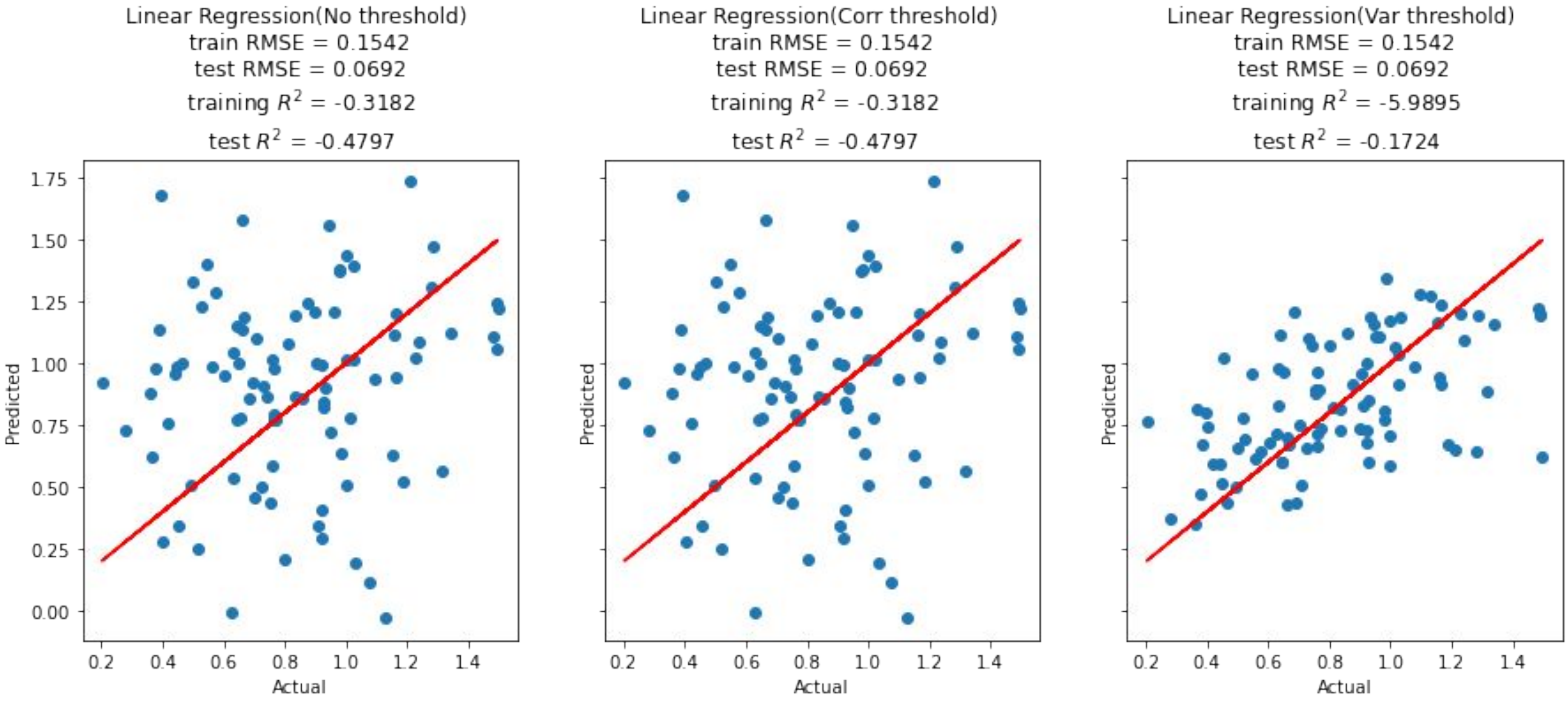


In the previous slide, a bar chart shows all the variables and their respective variances. Year has the highest variance, making it more important in determining Yield by this metric. Followed by LST_OCT and so on. As we increase the threshold, the lesser features we observe.

RandomForestRegressor



Linear Regression Model



RandomForestRegressor vs. Linear Regression

For both models there is not a discernible difference between before and after Correlation threshold feature selection. However, Random Forest overfits on all three occasions, as the test RMSE and R-squared values are poorer in test than in training.

The Linear Regression on the other hand, performs better in all occasions in as far as test RMSE is concerned. But, falls short on R-squared in both No Threshold and Corr Threshold scenarios. R-squared improves quite significantly in the Variance Threshold scenario. Making it the best model of the six.