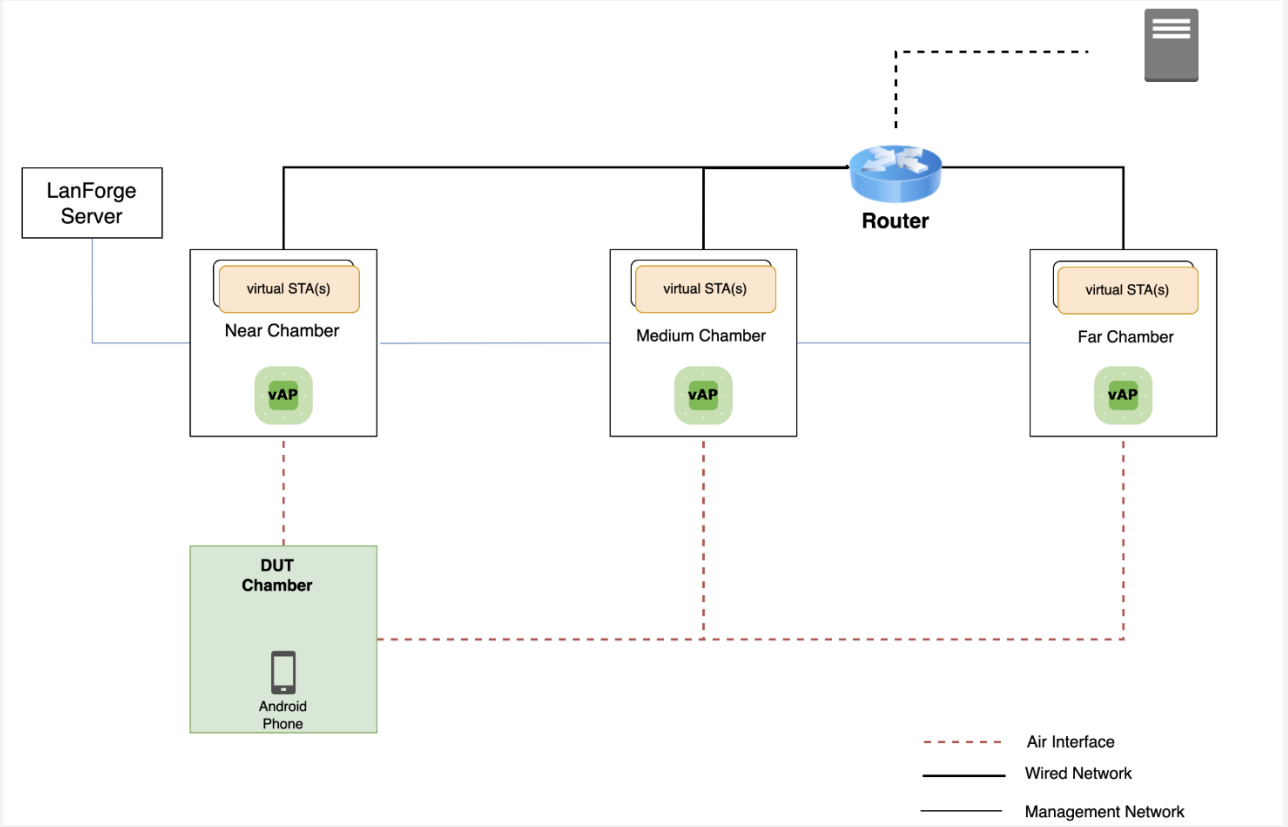
Predicting QoE based on WiFi KPIs

Arun Yerra

arunsarat@google.com

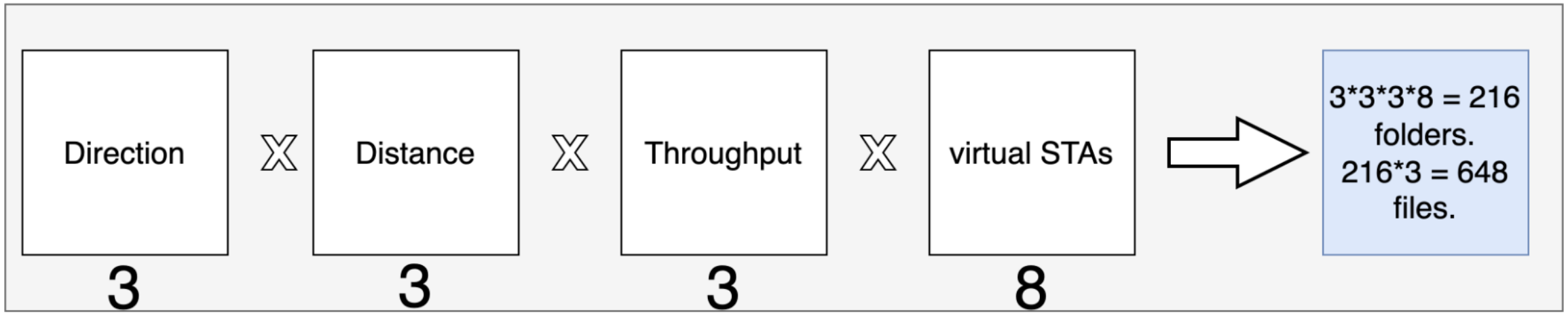
Introduction

- WiFi has been the go-to access network within a home environment. End users are more interested in user experience than speeds.
- The goal of the setup is to simulate a home environment, where a device can be at different distances from the AP and multiple devices connect to AP and contend for the air interface.
- Focus with this experimentation is about how end user quality of experience vary with respect to WiFi metrics.
- Our goals is to evaluate multiple ML/AI models in predicting end user Quality of Experience based the real-time WiFi and application metrics data.
- Setup includes 4 chambers; one houses the STA that is DUT and other three house virtual APs and virtual STAs in 3 different distances that simulate STAs that are near, far and medium distances.
- Number of virtual STAs in the chambers are changed from 1-8 to simulate devices contending for air resources in a typical home.
- Application bandwidth also changed between 10, 30 & 50 Mbps to simulate applications with varying bandwidth requirements.
- Traffic from client STA is routed through WiFi AP, WAN router to the cloud server and vice versa. Application server platform captures application QoE metrics and Candela LanForge server capture WiFi metrics.



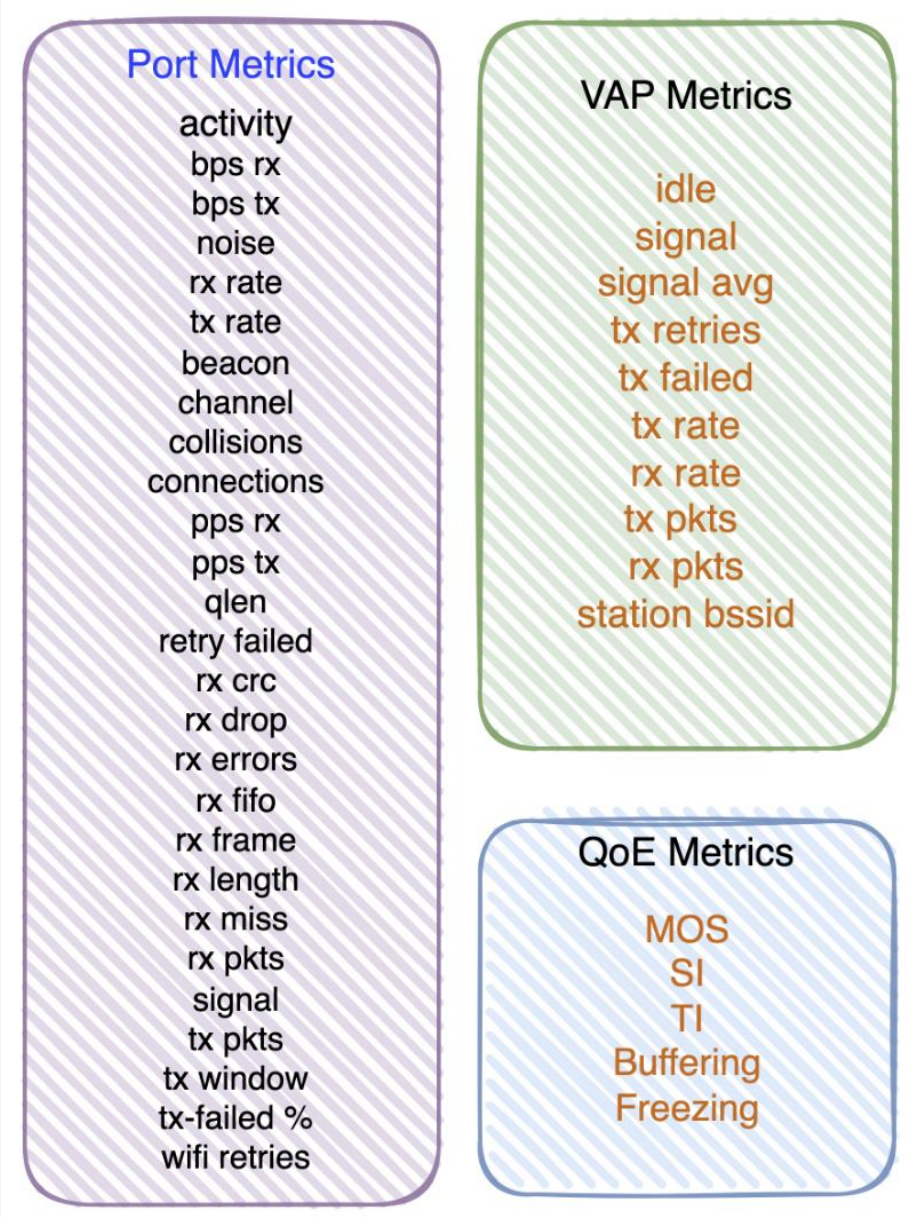
Data Pre-processing

- This experiment is conducted to simulate different conditions in a home environment.
- Hence the testing is repeated for the following combination of parameters:
  - Data traffic direction (downstream/upstream/both)
  - Number of devices (1-8)
  - Application throughput (10/30/50 Mbps)
  - Distance (Near, Medium, Far)
- There are around 216 folders each for combination of direction, distance, throughput and number of STAs.
- Each folder contains 3 files: port metrics, vap_metrics & QoE metrics. Hence, we've around 216 * 3 = 648 files.



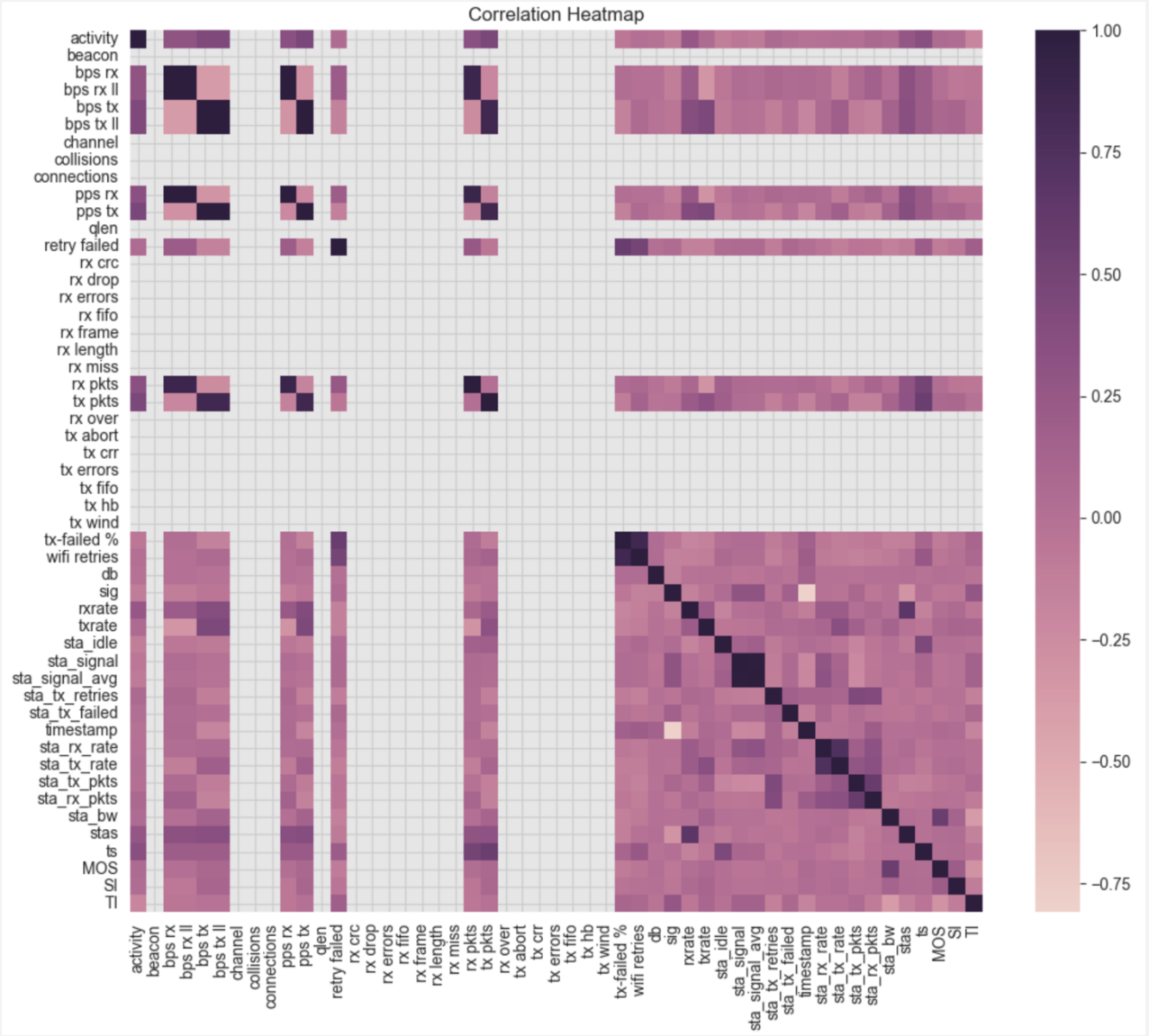
Understanding the data

- For each test scenario, video simulation lasts for a minute.
- Candela system captures WiFi metrics at every 0.5 seconds, while Spirent Umetrix software captures metrics at every 0.25 seconds.
- WiFi metrics are captured in two files: port metrics & vap_metrics.
  - Port metrics captures AP stats.
  - vSTA metrics captures traffic stats of virtual APs.
- Port metrics are duplicated to match virtual STA metrics captured based on the timestamp and then both those files are merged.
- Merging QoE metrics file is bit more complicated. One strategy is to get averages for each half second and then merge with port, virtual STA metrics.



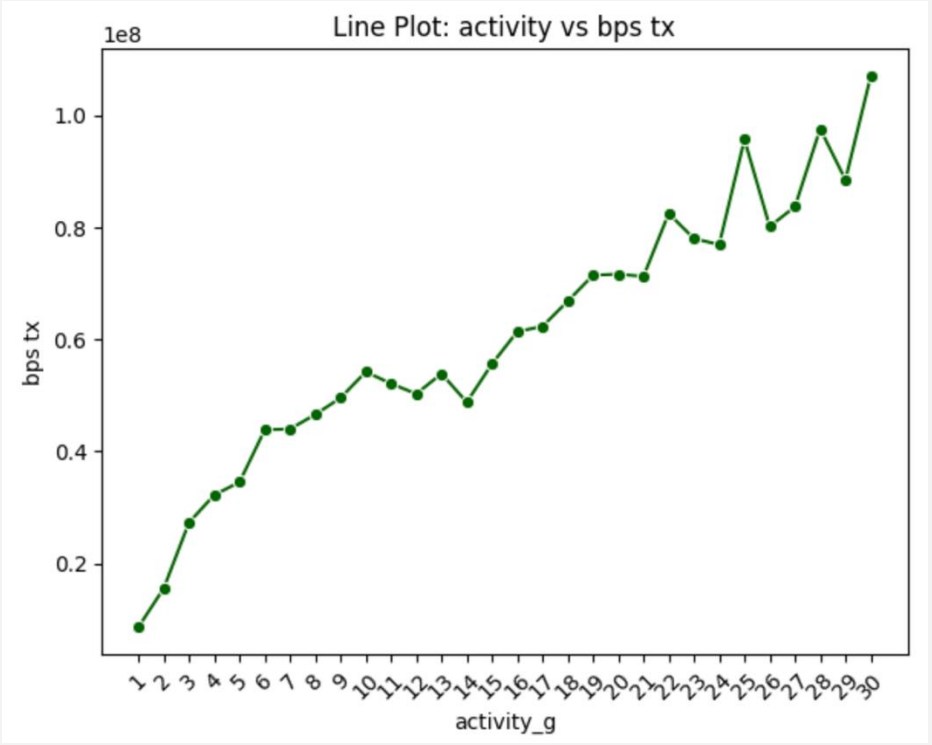
Feature Correlation

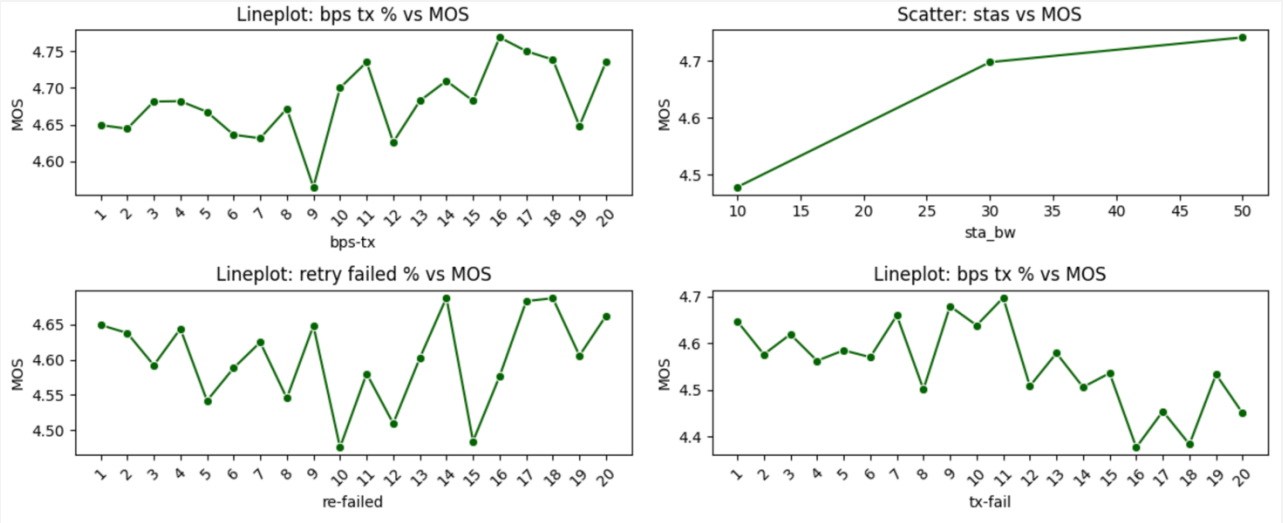
- Feature correlation is step in data pre-processing to understand the relationship between features.
  - Feature dimensionality reduction
  - Eliminate duplication features.
- In our WiFi metrics data, the following features have almost perfect correlation.
  - bps rx & bps rx II
  - bps tx & bps tx II
  - pps rx & bps rx
  - pps tx & bps tx
  - STA signal & STA signal Average
  - pps tx & tx pkts
  - pps rx & rx pkts
- Hence the following features are eliminated from the data.
  - 'bps rx ll', 'bps tx ll', 'pps rx', 'pps tx', 'sta_signal', 'rx pkts', 'tx pkts'



Feature Correlation

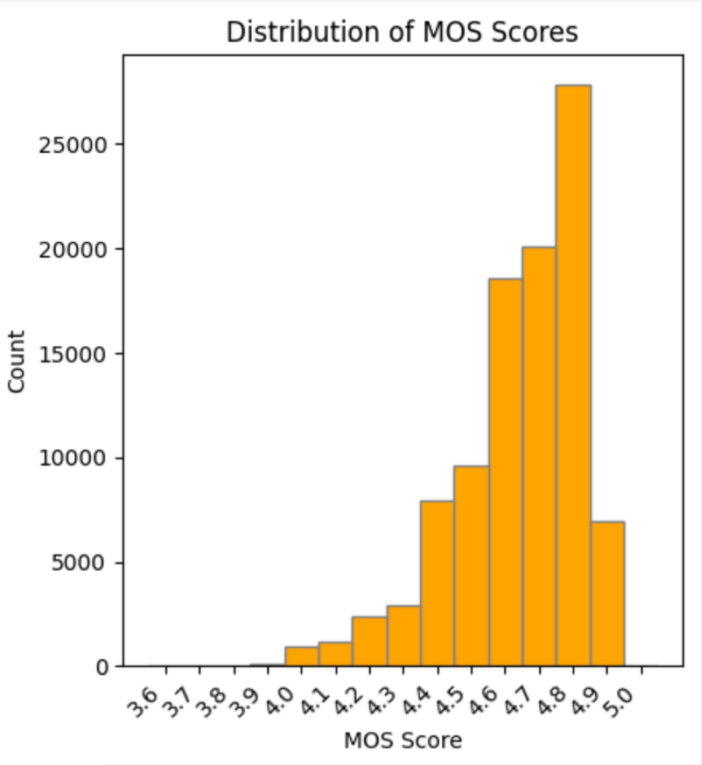
- Correlation Matrix shows that the following features are correlated highly, but not perfectly.
  - WiFi retries & tx-failed %
  - STA tx rate & STA rx rate
  - STA tx pkts & STA rx pkts
  - Tx-failed % & retry failed
  - MOS & STA throughput
- The following features are related most to MOS score (our target).
  - STA throughput
  - Bps tx
  - Tx-failed %
  - Retry-failed
- As diagram shows, the correlation with MOS score is not too strong.

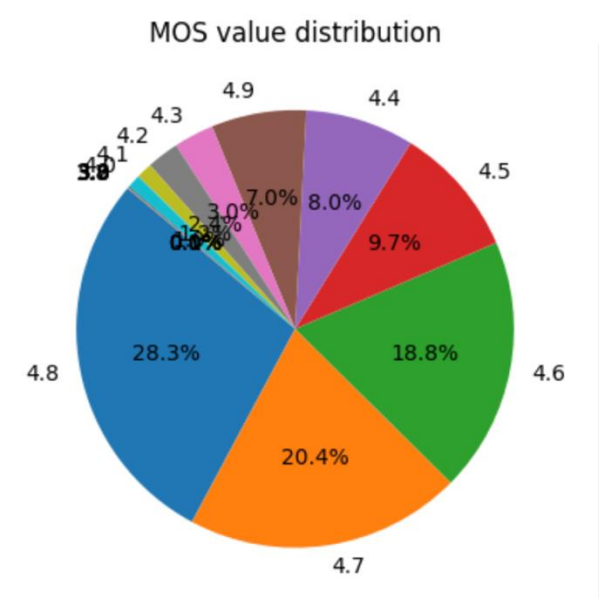




MOS Score Distribution

- In a data set where target values aren't distributed evenly, randomly selecting the training and testing data couldn't accurately represent the actual scenario.
- In our data set, MOS scores aren't distributed evenly.
  - MOS scores of 4.6 – 4.8 comprises about 60% of the overall MOS score distribution.
- For better accuracy of the regression models, we need to choose training and testing data set with similar target distribution as that of all the data.





Data Analysis

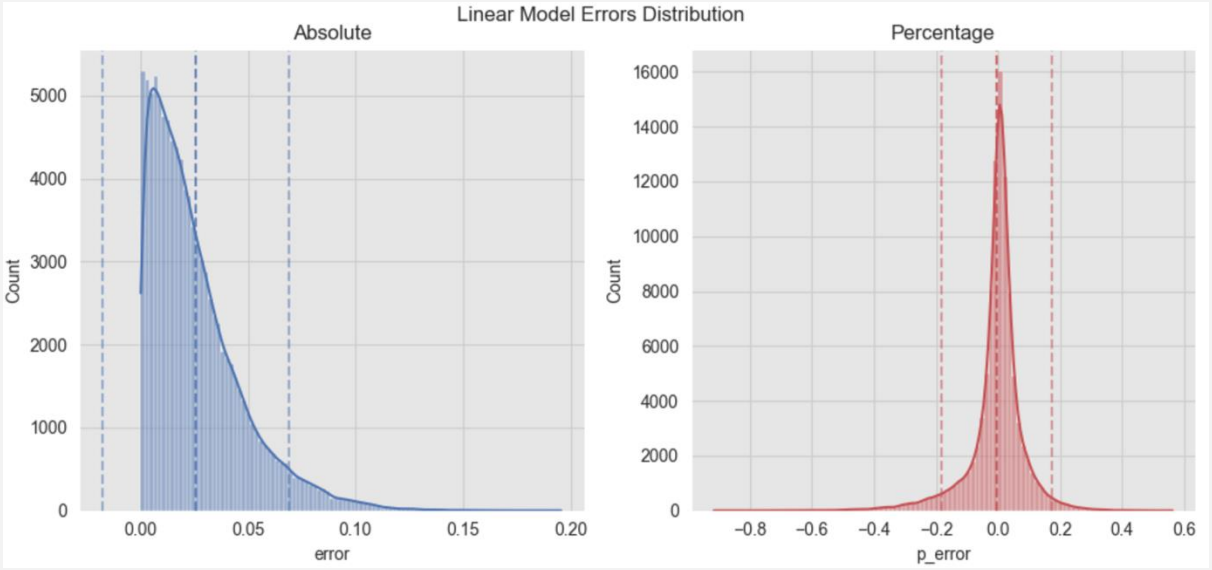
- Data analysis step for regression problems includes identifying multiple regression models and evaluating the performance and accuracy of them.
- With each model, we train the model with training data set and the evaluate the accuracy of the model's predicted values vs. actual test values.
- For each model that is evaluated, the following KPIs are measured and these KPIs are used to evaluate the efficacy and performance of a Model.
  - Average Error
  - Average Error percentage
  - Error Standard Deviation
  - Average Error percentage Standard Deviation.
  - Duration
- We evaluate the following models
  - Linear Regression with PCA
  - Linear Regression
  - Ridge Regression
  - Lasso Regression
  - K-nearest neighbors Regression
  - Decision Tree
  - SVC regression

Key	Description
model	Model Name
mean_error	Mean Error
mean_perror	Mean Percentage Error
std_error	Error Stanadard Deviation
std_perror	Percentage Error Standard Deviation
duration	Duration

Linear Regression with PCA components

- PCA identifies principal components which captures the features that impact most of the variance.
  - PCA transforms original features into uncorrelated features.
  - PCA eliminates correlated features and less informative.
- Linear regression performs well with less correlated input features and PCA generates these principal components.
- This model has mean error of 0.03 and has -0.48% mean percentage of error.
- Error standard deviation is at 0.02 and percentage error standard deviation of 8.98%.
- It took about 11 seconds for training this model.

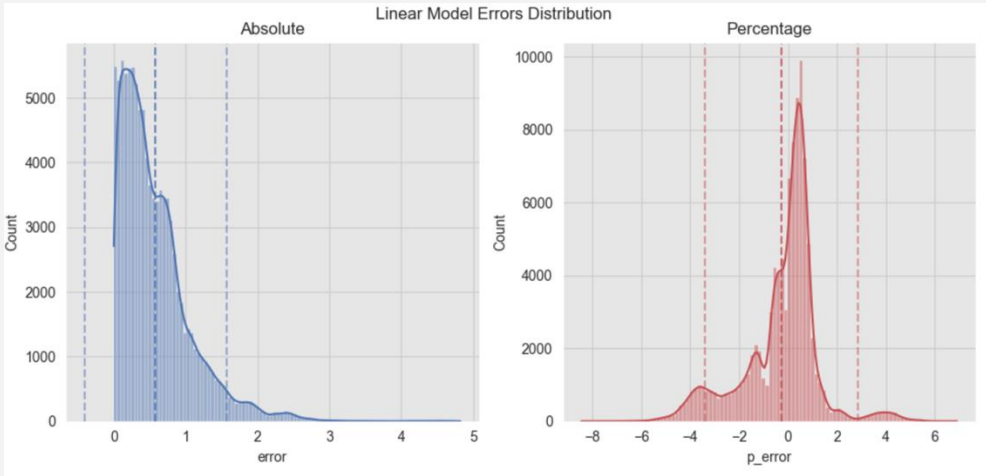
```
## Regressor: LinearRegression() Training Report ##
Training time:  10.57568097114563 seconds
Best Parameters:  {'poly__degree': 2, 'regressor__fit_intercept': True}
Best training Accuracy: 0.9988755629728281
Test Set Accuracy: 0.998864302308859
error mean =  0.03, error std =  0.02
p_error mean = -0.48%, p_error std =  8.98%
```



Linear Regression

- Linear Regression model assumes linear relation between the input features and the data training step identifies a linear relation between the features.
- Model pipeline has the following components.
  - Standard Scaler
  - Polynomial Feature Transformation
  - Linear Regression
- This model has mean error of 0.58 and has -27.48% mean percentage of error.
- Error standard deviation is at 0.49 and percentage error standard deviation of 156.23%.
- It took about 48 seconds for training this model.

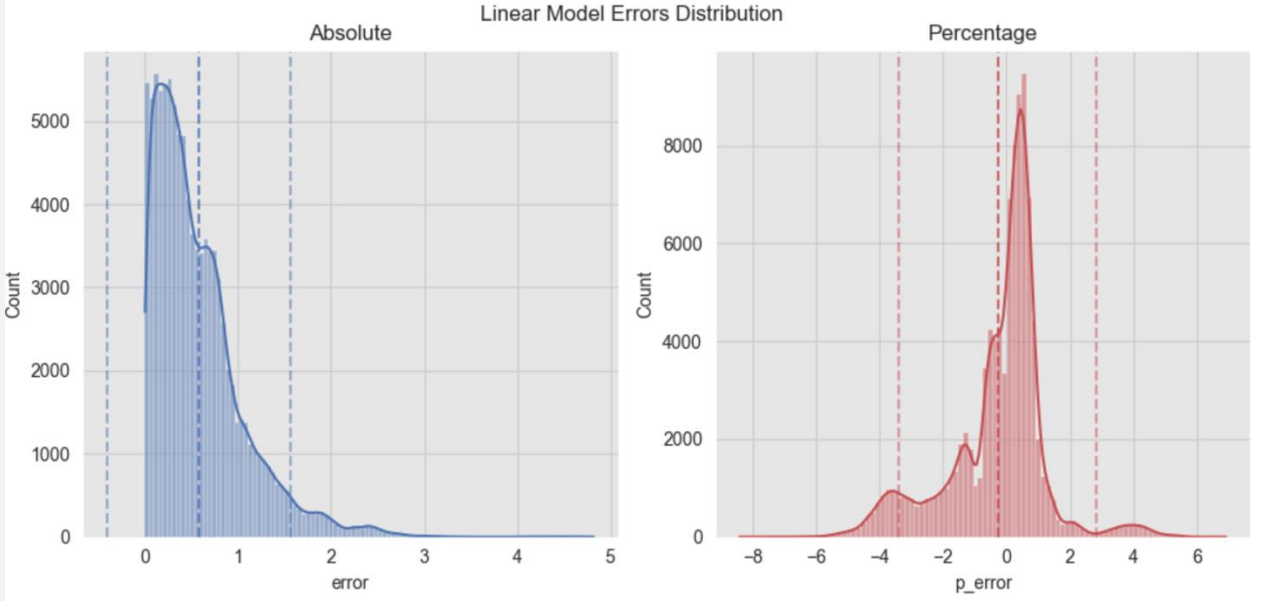
```
## Regressor: LinearRegression() Training Report ##
Training time:  47.68888711929321 seconds
Best Parameters:  {'poly__degree': 2, 'regressor__fit_intercept': True}
Best training Accuracy: 0.42239031676825045
Test Set Accuracy: 0.41965364262471483
Best Parameters: {'poly__degree': 2, 'regressor__fit_intercept': True}
Best Cross-Validated MSE: -0.42239031676825045
error mean =  0.58, error std =  0.49
p_error mean = -27.48%, p_error std =  156.23%
```



Ridge Regression

- Ridge regression is linear regression with a penalty component which diminishes the impact of outliers on the target.
- This penalty term constrains the size of the coefficients (weights), encouraging the model to learn simpler and more robust parameters.
- Model pipeline has the following components.
  - Standard Scaler
  - Polynomial Feature Transformation
  - Ridge Regression
- This model has mean error of 0.58 and has -27.49% mean percentage of error.
- Error standard deviation is at 0.49 and percentage error standard deviation of 156.23%.
- It took about 54 seconds for training this model.

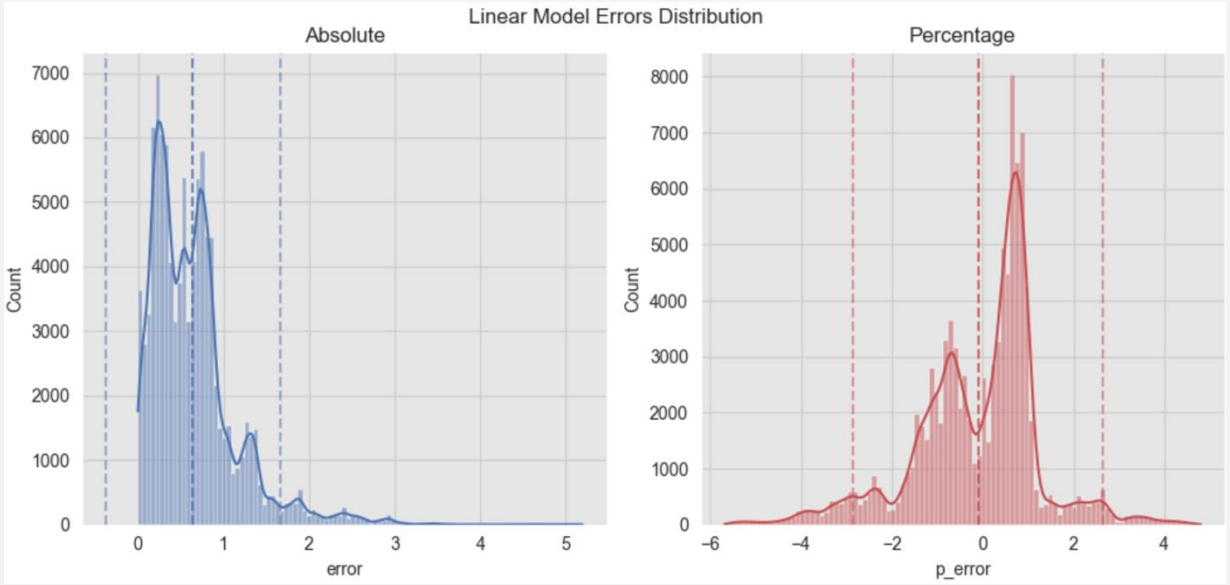
```
## Regressor: Ridge(alpha=0.1) Training Report ##
Training time:  53.611799240112305 seconds
Best Parameters:  {'poly__degree': 2, 'regressor__alpha': 0.1, 'regressor__fit_intercept': True}
Best training Accuracy: 0.42238983309441835
Test Set Accuracy: 0.41963258208764564
Best Parameters: {'poly__degree': 2, 'regressor__alpha': 0.1, 'regressor__fit_intercept': True}
Best Cross-Validated MSE: -0.42238983309441835
error mean =  0.58, error std =  0.49
p_error mean = -27.49%, p_error std =  156.23%
```



Lasso Regression

- Lasso regression is a variation of linear regression with inclusion of L1 regularization
- Lasso can automatically select the most important features by shrinking the coefficients of less important features to zero
- Lasso helps prevent overfitting and improves model generalization and it also can select the most important features by reducing the coefficient of less important features to zero.
- Model pipeline has the following components.
  - Standard Scaler
  - Polynomial Feature Transformation
  - Lasso Regression
- This model has mean error of 0.64 and has -9.63% mean percentage of error.
- Error standard deviation is at 0.51 and percentage error standard deviation of 137.82%.
- It took about 70 seconds for training this model.

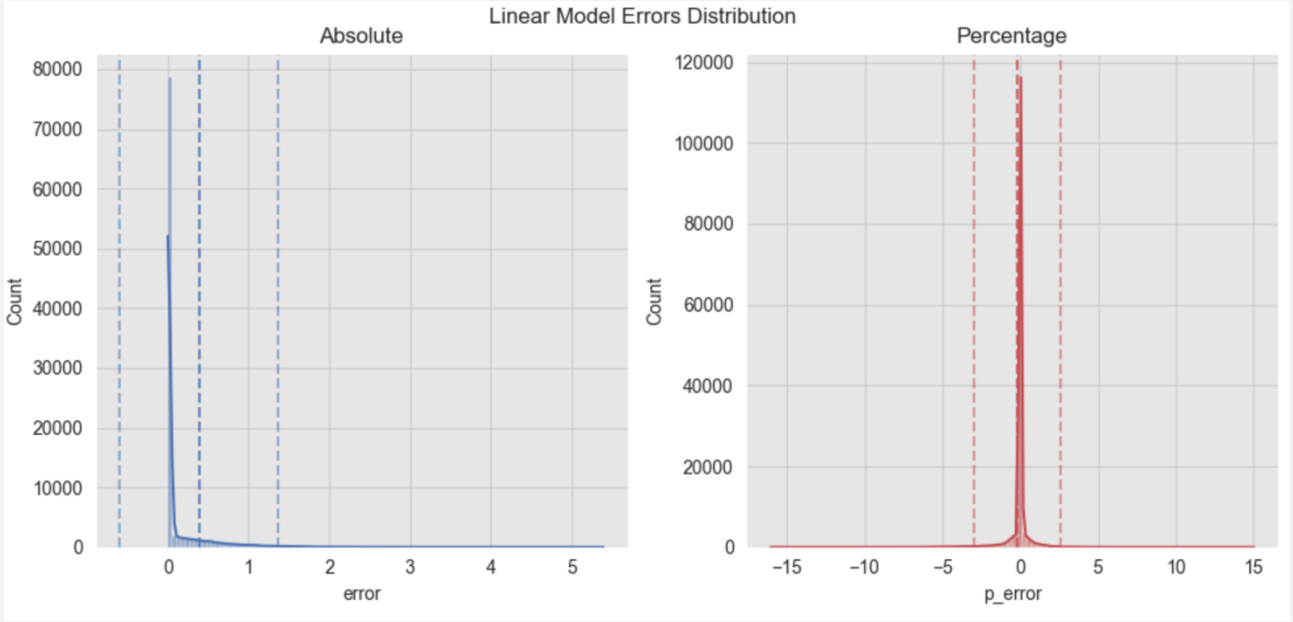
```
## Regressor: Lasso(alpha=0.1) Training Report ##
Training time:  70.62069201469421 seconds
Best Parameters:  {'poly__degree': 2, 'regressor__alpha': 0.1, 'regressor__fit_intercept': True}
Best training Accuracy: 0.32934102619345246
Test Set Accuracy: 0.3276700240345298
Best Parameters: {'poly__degree': 2, 'regressor__alpha': 0.1, 'regressor__fit_intercept': True}
Best Cross-Validated MSE: -0.32934102619345246
error mean =  0.64, error std =  0.51
p_error mean = -9.63%, p_error std =  137.82%
```



K-nearest neighbor Regression

- KNN predicts the target value by computing average of the target values of the surrounding neighbors.
- KNN is non-parametric unlike linear regression, because there's no specific form of the data.
- Model pipeline has the following components.
  - Standard Scaler
  - knn regression
- This model has mean error of 0.39 and has -18.98% mean percentage of error.
- Error standard deviation is at 0.49 and percentage error standard deviation of 139.31%.
- It took about 2 minutes 36 seconds for training this model.

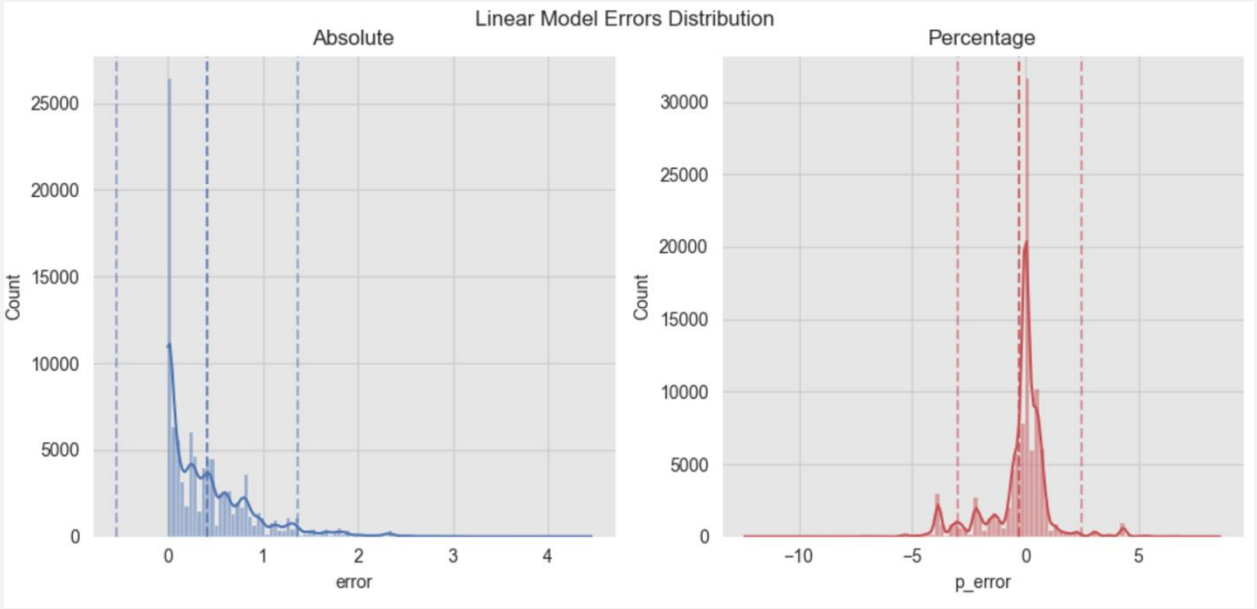
```
## Regressor: KNeighborsRegressor(p=1, weights='distance') Training Report ##
Training time:  156.29849290847778 seconds
Best Parameters:  {'regressor__n_neighbors': 5, 'regressor__p': 1, 'regressor__weights': 'distance'}
Best training Accuracy: 0.5963191609420394
Test Set Accuracy: 0.6086132090882999
Best Parameters: {'regressor__n_neighbors': 5, 'regressor__p': 1, 'regressor__weights': 'distance'}
Best Cross-Validated MSE: -0.5963191609420394
error mean =  0.39, error std =  0.49
p_error mean = -18.98%, p_error std =  139.31%
```



Decision Tree Regression

- Decision Tree builds tree like structure to model relationship between input features and target variable.
- The dataset is split recursively based on the feature values into subsets, and each split aims to minimize the variance of the target variable in the resulting subsets.
- The final prediction is made by averaging the target values in the leaves of the tree.
- Model pipeline has the following components.
  - Standard Scaler
  - Decision Tree regression
- This model has mean error of 0.41 and has -25.41% mean percentage of error.
- Error standard deviation is at 0.48 and percentage error standard deviation of 137.17%.
- It took about 90 seconds for training this model.

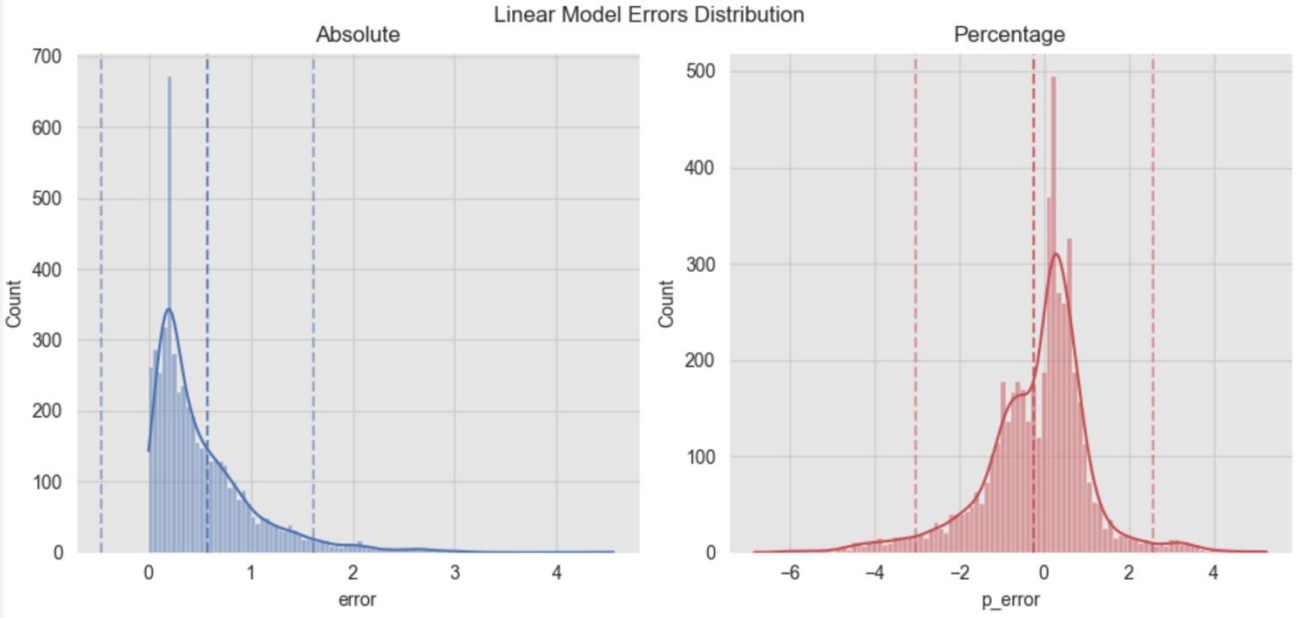
```
## Regressor: DecisionTreeRegressor(max_depth=10, random_state=42) Training Report ##
Training time:  89.11216998100281 seconds
Best Parameters:  {'regressor__max_depth': 10, 'regressor__max_features': None, 'regressor__min_samples_leaf': 1, 'regressor__min_samples_split': 2}
Best training Accuracy: 0.5944657643926682
Test Set Accuracy: 0.5990119708116548
Best Parameters: {'regressor__max_depth': 10, 'regressor__max_features': None, 'regressor__min_samples_leaf': 1, 'regressor__min_samples_split': 2}
Best Cross-Validated MSE: -0.5944657643926682
error mean =  0.41, error std =  0.48
p_error mean = -25.41%, p_error std =  137.17%
```



SVM Regression

- SVR is designed to fit the best possible line (or hyperplane) within a margin of tolerance, allowing for a certain amount of error while trying to maximize the margin.
- In SVR, we define a margin of tolerance (ε) within which predictions are not penalized. The goal is to find a function that fits within this margin while keeping as many data points as possible within the margin.
- Model pipeline has the following components.
  - Standard Scaler
  - SVR regression
- This model has mean error of 0.58 and has -23.26% mean percentage of error.
- Error standard deviation is at 0.52 and percentage error standard deviation of 140.17%.
- SVR has heavy compute requirements. Hence, we took a sample of 5000 data points (about 5% of the total data set).
- Even with the sampled data, it took 6 minutes of training with SVR model. Basically, SVR compute requirements is disproportionally high compared to other models.

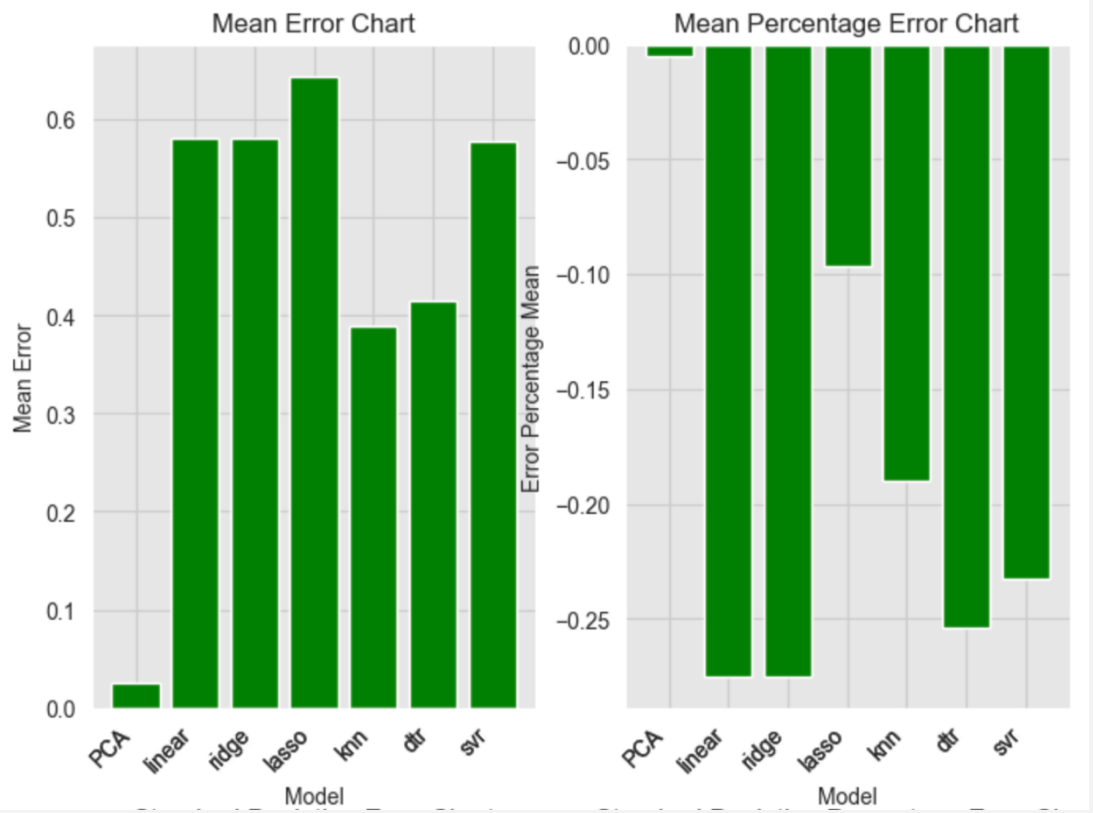
```
## Regressor: SVR(C=10, epsilon=0.2) Training Report ##
Training time:  366.04166197776794 seconds
Best Parameters:  {'regressor__C': 10, 'regressor__epsilon': 0.2, 'regressor__gamma': 'scale', 'regressor__kernel': 'rbf'}
Best training Accuracy: 0.36525740628206294
Test Set Accuracy: 0.3849863124547691
Best Parameters: {'regressor__C': 10, 'regressor__epsilon': 0.2, 'regressor__gamma': 'scale', 'regressor__kernel': 'rbf'}
Best Cross-Validated MSE: -0.36525740628206294
error mean =  0.58, error std =  0.52
p_error mean = -23.26%, p_error std =  140.33%
```



Conclusion

- To evaluate the model performance and accuracy, we plotted the following:
  - Mean error
  - Mean percentage of Error
  - Error standard deviation
  - Percentage of error STD
- Linear regression and variations of it like Ridge regression and Lasso regression have similar mean error, SVR also has similar mean error.
- Linear regression with PCA has the lowest mean error and lowest mean percentage of error.
- KNN and decision tree regression provide a marginal improvement than linear regression variations.

	model	mean_error	mean_perror	std_error	std_perror	duration
0	PCA	0.025665	-0.004777	0.021768	0.089784	8.669481
1	linear	0.580630	-0.274794	0.493360	1.562329	45.513772
2	ridge	0.580667	-0.274919	0.493338	1.562307	52.407080
3	lasso	0.644111	-0.096332	0.507612	1.378215	61.596516
4	knn	0.388700	-0.189845	0.490336	1.393122	271.134386
5	dtr	0.414277	-0.254147	0.479057	1.371743	108.655557
6	svr	0.577539	-0.232608	0.522010	1.403253	383.900782



Conclusion

- Standard deviation of error and percentage of error is considerably less for linear regression with PCA than rest of the other models.
- And duration for training is also considerably less for linear regression with PCA components.
- SVR even with just 5% sampled data consumes a lot of compute and takes around 6 minutes, while linear regression of PCA components take 11 seconds.
- Hence, I conclude that linear regression with PCA has the best accuracy in predicting the user QOE MOS score, given WiFi metrics collected in home.
- But with Linear regression with PCA requires the future in put data to be transformed into PCA components before passing them as input.