Improving Predictive Accuracy through Correlation-Based Data Frame Splitting for Polynomial and Linear Models

Mr Atul Rajesh Waman[1], Dr.Sangeeta Mahesh Borde[2]
[1,2]*MIT Arts, Commerce and Science College, Alandi(D), Pune, Maharashtra*

Abstract- This paper presents a method to improve the accuracy of polynomial and linear models by splitting a dataset into smaller, specialized sub-data frames based on the correlation between the target variable and its highest correlated feature. The process involves plotting this relationship, sorting the data by the highest correlated feature, and using the mode or visual observation to identify split points. The resulting sub-data frames are used to create simpler models that better fit their respective regions. During prediction, the appropriate sub-model is selected based on the value of the highest correlated feature, leading to more accurate and context-specific predictions.

Keywords- Regional modeling, correlation-based splitting, sub-data frames, predictive accuracy, and mode-based split.

1 INTRODUCTION

This paper will explore how Linear and Polynomial regression models can have better predictions and accuracy with the help of splitting our data set with respect to the highest correlated variable. This paper shows that overall fitted models (model trained on all data) have a higher percentage error than small mode or regional mode (model trained on just a chunk of all data).
Thus small models have better accuracy. One may follow the given steps and see for themselves that accuracy increases.

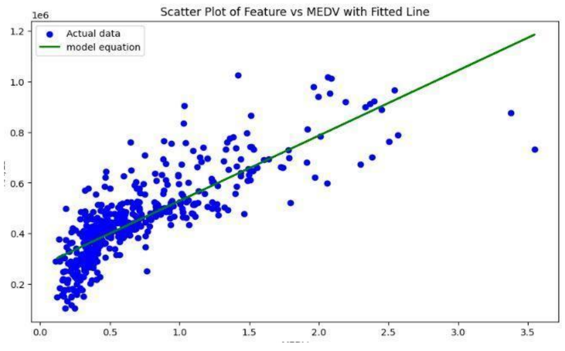
Linear Regression Model [1]
Linear Regression models are is the simplest regression model to train and it has its own benefits and advantages. Linear regression models fit a straight line to the data, which is good when the data follows a linear pattern on a scatter diagram. Linear Regression models the relationship between the dependent variable (called as target or label) and one or many independent variables (called as features). The simple equation of the linear regression model with one feature ‘x’ can be given as

predication = $F_{w,\,b}$ (x) = w.x + b

Here, ‘x’ is feature and ‘w’ is the relevant value to adjust x or slope of line and ‘b’ is y-axis intercept or the point where line interacts with y-axis. The same equation for multiple features can be given by

predication = $F_{w1\,,w2\,,w3\,,\,\ldots\,,b}$ (x1,x2,x3..) = w1.x1+ w2.x2 + w3.x3 + … + b

Fallowing is the graph of the Linear Regression model with the scatter diagram of Housing data set.



Disadvantages of Linear Regression

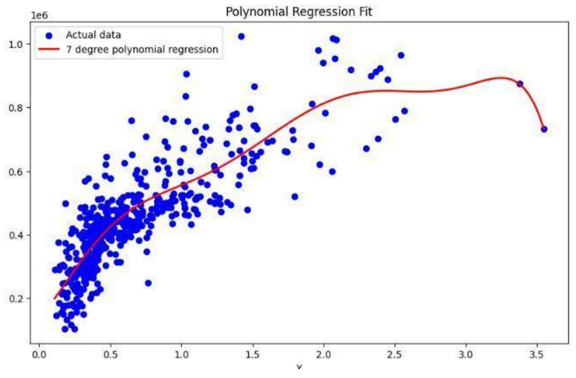
1. Assumes Linearity: One of the main limitations is the assumption of a linear relationship between the dependent and independent variables. If the actual relationship is non-linear, linear regression may not provide a good fit.
2. Sensitive to Outliers: Linear regression is highly sensitive to outliers, which can significantly affect the slope of the line and lead to misleading results.
3. Cannot be fit to curvy data: fitting a linear model on little curvy data will cause significant error in prediction.

4. Limited to Linear Relationships: Linear regression is limited to modelling linear relationships, so it cannot capture more complex patterns in data without transformation or use of other techniques.

Polynomial Model

[1] Polynomial regression is an extension of linear regression that models the relationship between the independent variable x and the dependent variable y as an nth-degree polynomial. Unlike linear regression, which assumes a straight-line relationship, polynomial regression can model non-linear relationships by introducing polynomial terms of the independent variable. The general form of a polynomial regression model is:

[3] $y = F_{w1, w2, ..., b}(x, x^2, x^3, ..) = w1.x + w2.x^2 + w3.x^3 + \ldots + b$



Disadvantages of Polynomial Regression

1. Risk of Over fitting: A high-degree polynomial can fit the training data too closely, capturing noise and outliers, leading to poor generalization on unseen data.
2. Complexity: As the degree of the polynomial increases, the model becomes more complex, making it harder to interpret and more computationally expensive.
3. Sensitive to Outliers: Similar to linear regression, polynomial regression is sensitive to outliers, which can disproportionately affect the curve and lead to misleading results.

Research Gap & Problem Formulation:

In my paper and the [5]"Model Soups" paper differ fundamentally in their approaches to improving model accuracy. My paper emphasizes segmenting the dataset into smaller, specialized sub-dataframes based on the most correlated variable, and then training individual models on these subsets to enhance predictive accuracy by selecting the most relevant model for each prediction. In contrast, the [5] "Model Soups" paper explores averaging the weights of multiple fine-tuned models with varying hyper parameters to create a single, more robust model. This approach aims to improve overall model performance without increasing inference time, but does not involve data segmentation or dynamic model selection based on specific input characteristics.

The clear problem with Linear and Polynomial model is not giving better accuracy on curvy data. It does not fit curvy data properly or in case of polynomial model, it may get overfit while trying to have better fit.

I have found that by splitting data into smaller parts we can achieve better accuracy. This has not been done in any other paper before.

1.5 Objective

Through this paper we want to achieve better fit and accuracy in Linear and Polynomial models by introducing idea of regional model fitting.

To achieve this we will perform many measures and steps such as split data, avoiding overfitting and getting better accuracy.

2. LITERATURE REVIEW

1. Traditional regression models like linear and polynomial regression often struggle with curvilinear data, leading to issues like overfitting or inadequate model fit.
2. Recent research has explored methods to enhance predictive accuracy by fine-tuning model parameters and using advanced techniques such as "model soups," which average the weights of multiple fine-tuned models to improve performance without additional inference costs [5].
3. This research paper builds on this by introducing a novel approach of correlation-based splitting of dataframes, which creates region-specific models. This method has been shown to reduce errors and improve accuracy compared to models trained on entire datasets in my paper.
4. The approach aligns with recent trends in improving model robustness and accuracy through

innovative data handling and model training techniques.

3. METHODOLOGY

Techniques and method used this paper are very simple and handy to implement. Data set used is in this project is available on Kaggle.

Data set is of Housing, which is easy and simple to understand. Housing data set have four major fields like RM, LSTAT, PTRATIO and MEDV. You will get description of data very easily with data set on Kaggle. Our motive behind understanding data fields is to understand correlation. I have used feature engendering to construct better correlated data fields like RM/LS which is nothing but RM divided by LSTAT.

We are going to use basic Sklearn which is frame work for machine learning and is really helpful when it comes to using models like Linear and Polynomial regression, train - test - split and error and accuracy finding. Apart from Sklearn, we would require Pandas and Numpy for data storing and preprocessing. Pandas gives us various features for storing and manipulating dataset in forms of data frames.

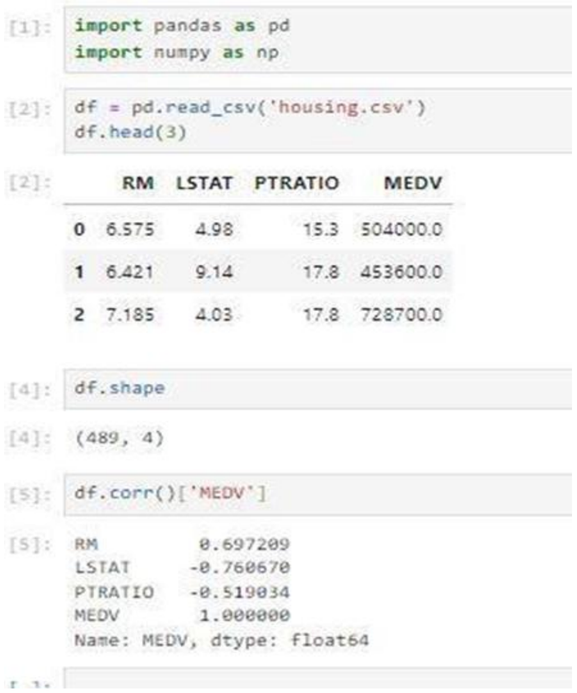
At end few algorithms used may depend upon what you want to do with data and how are you approaching problem. We may use gradient descent algorithm if we want to get best value for our w1, w2, … which are coefficient's of features x1, x2, ...

for setting up for experiment one would need Jupyter Notebook or Google Colab. Python and pip, Pandas and Numpy with Sklearn installed.

4. EXPERIMENT

This section will cover the road map of how to actually implement this paper with all the code snippets required for importing data, processing and also model training.

Download housing data set or any other data set from Kaggle. Install pandas and Numpy in Jupyter notebook. Start from importing required libraries likes pandas and Numpy. Check shape of data and also check fields. I have taken housing data set and downloaded .csv file from Kaggle with RM, LSTAT, PTRATIO and MEDV. Here MEDV is the price of the house. To implement this paper we don't have to think a lot about the data fields but we just have to be concern with correlation.



```
[1]: import pandas as pd
     import numpy as np

[2]: df = pd.read_csv('housing.csv')
     df.head(3)
```

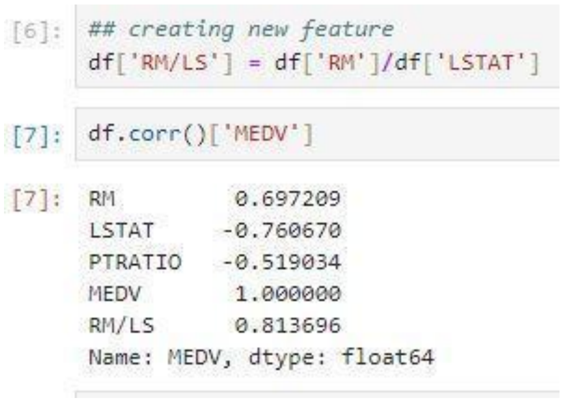
[2]:

	RM	LSTAT	PTRATIO	MEDV
0	6.575	4.98	15.3	504000.0
1	6.421	9.14	17.8	453600.0
2	7.185	4.03	17.8	728700.0

```
[4]: df.shape

[4]: (489, 4)

[5]: df.corr()['MEDV']

[5]: RM          0.697209
     LSTAT      -0.760670
     PTRATIO    -0.519034
     MEDV        1.000000
     Name: MEDV, dtype: float64
```

We have taken correlation with respect to MEDV as it is target or label and we would want our target to increase or decrease with respect to other features.

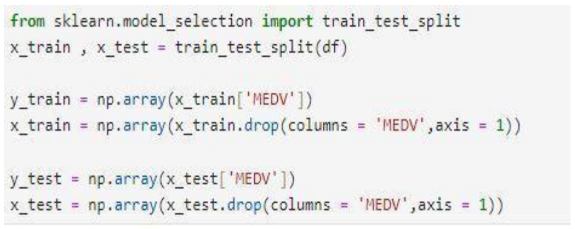
Feature Engineering:
we can compose new feature i.e. RM/LSTAT which have high correlation. This need not come to mind intuitionally but by plotting graph of all features with all other features.



```
[6]: ## creating new feature
     df['RM/LS'] = df['RM']/df['LSTAT']

[7]: df.corr()['MEDV']

[7]: RM          0.697209
     LSTAT      -0.760670
     PTRATIO    -0.519034
     MEDV        1.000000
     RM/LS       0.813696
     Name: MEDV, dtype: float64
```

As you can see in code snippet, RM/LS have best positive correlation.

Training model for overall data:
Train test split:
Now we have to train a model for overall data or big data frame. After this we will also train model for smaller data frame after splitting data for regional model.
First step is to do train-test-split of our big data frame. We have to first import Sklearn for that and work use train test split from sklearn.model_selection.



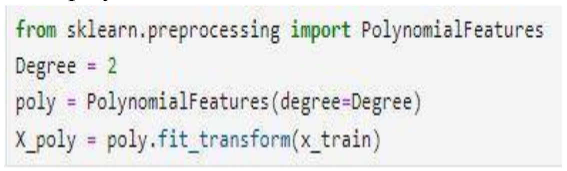
```
from sklearn.model_selection import train_test_split
x_train , x_test = train_test_split(df)

y_train = np.array(x_train['MEDV'])
x_train = np.array(x_train.drop(columns = 'MEDV',axis = 1))

y_test = np.array(x_test['MEDV'])
x_test = np.array(x_test.drop(columns = 'MEDV',axis = 1))
```

Creating Polynomial features:

[2] We need to create polynomial features of degree 2 or more according to complexity of our data. Following snippet provides code to do that. We will get X □ X^2 in degree 2. We will take 'degree' as variable, this will help us to quickly modify the value of degree and generate polynomial features.

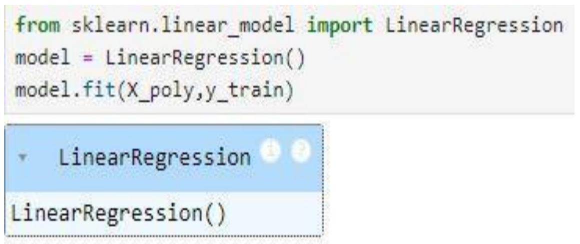
As you can see we have got array of Quadratic features because of degree 2 for all four features.

Train polynomial model:



```
from sklearn.preprocessing import PolynomialFeatures
Degree = 2
poly = PolynomialFeatures(degree=Degree)
X_poly = poly.fit_transform(x_train)
```

We would need linear regression model and we will then feed it with polynomial features to train it.

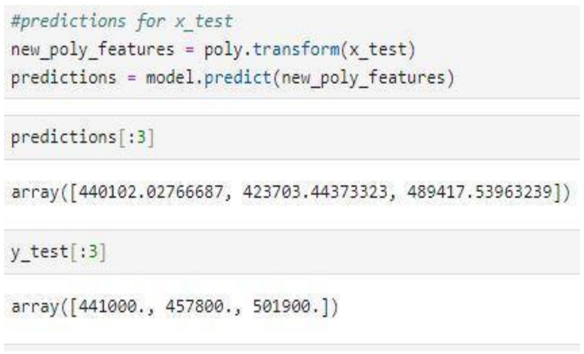


```
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(X_poly,y_train)
```

```
▾ LinearRegression
LinearRegression()
```

Finding Predications:

Use test to find predications and later on calculate accuracy.



```
#predictions for x_test
new_poly_features = poly.transform(x_test)
predictions = model.predict(new_poly_features)
```

```
predictions[:3]
```

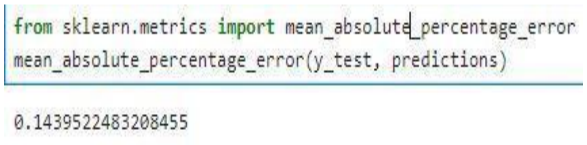
```
array([440102.02766687, 423703.44373323, 489417.53963239])
```

```
y_test[:3]
```

```
array([441000., 457800., 501900.])
```

Finding percentage error for x_test:

We can use mean_absolute_percentage_error.

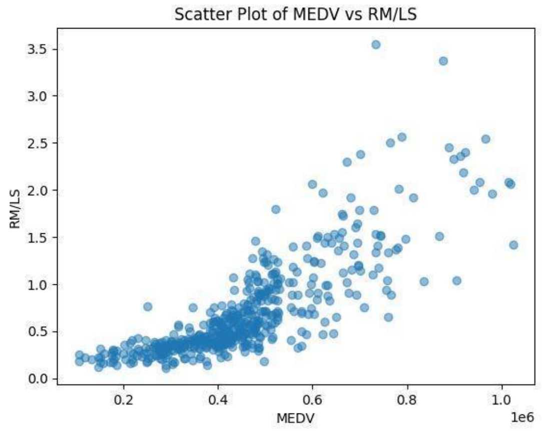


```
from sklearn.metrics import mean_absolute_percentage_error
mean_absolute_percentage_error(y_test, predictions)
```

```
0.1439522483208455
```

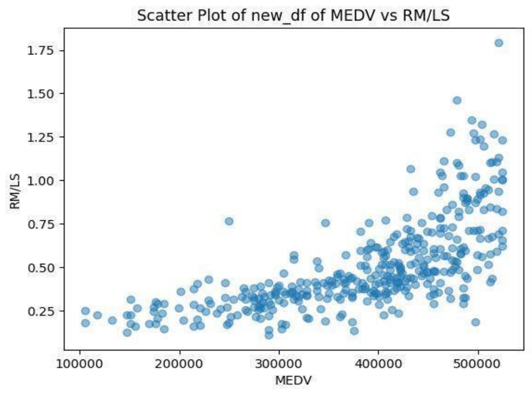
This was complete process for creating model and getting predictions.

Mode Based Splitting:
In this section we need to split our data frame into smaller data frame based on mode of data.
Fallowing is the scatter diagram of original data frame or big data set.



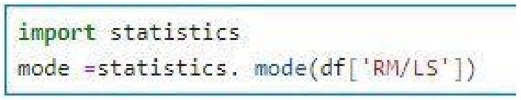
Now, we want to achieve splitting of data based on mode. This can be done with visual method or with consideration to mode.

We will get this portion of above graph:



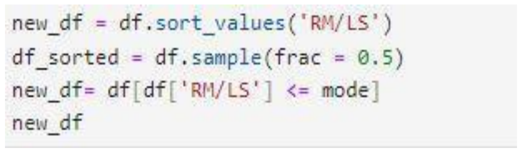
Fallow the following steps to achieve this:

1. Calculate mode of highest correlated variable. For this we may use statistics library.



```
import statistics
mode =statistics. mode(df['RM/LS'])
```

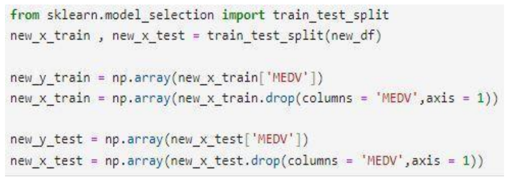
2. Sort data frame with respect to most correlated variable:



```
new_df = df.sort_values('RM/LS')
df_sorted = df.sample(frac = 0.5)
new_df= df[df['RM/LS'] <= mode]
new_df
```

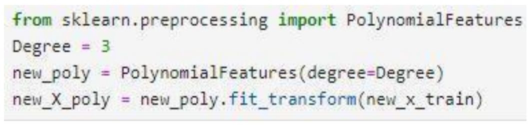
this will give us new_df which consist of data points from RM/LS equal to 0 to mode.

3. Train test split:
here we have to do same process that we did before. I just have added same code but with different



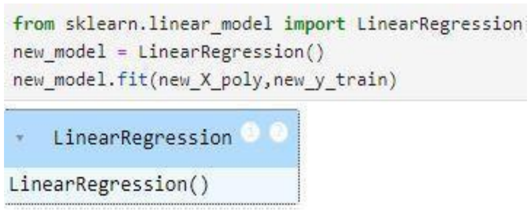
```
from sklearn.model_selection import train_test_split
new_x_train , new_x_test = train_test_split(new_df)

new_y_train = np.array(new_x_train['MEDV'])
new_x_train = np.array(new_x_train.drop(columns = 'MEDV',axis = 1))

new_y_test = np.array(new_x_test['MEDV'])
new_x_test = np.array(new_x_test.drop(columns = 'MEDV',axis = 1))
```

4. Find suitable degree and create polynomial features:
I found out that for my data and small data frame degree = 3 fits well and gives least error. So I used degree = 3.



```
from sklearn.preprocessing import PolynomialFeatures
Degree = 3
new_poly = PolynomialFeatures(degree=Degree)
new_X_poly = new_poly.fit_transform(new_x_train)
```

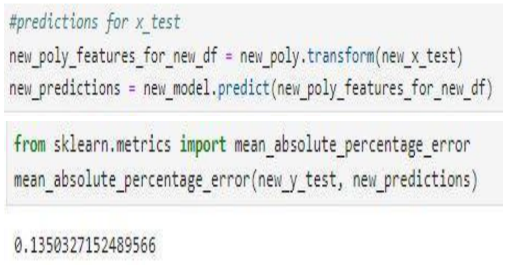
5. Training model:



```
from sklearn.linear_model import LinearRegression
new_model = LinearRegression()
new_model.fit(new_X_poly,new_y_train)
```

```
LinearRegression
LinearRegression()
```

6. Getting predictions and finding percentage error:



```
#predictions for x_test
new_poly_features_for_new_df = new_poly.transform(new_x_test)
new_predictions = new_model.predict(new_poly_features_for_new_df)
```

```
from sklearn.metrics import mean_absolute_percentage_error
mean_absolute_percentage_error(new_y_test, new_predictions)
```

```
0.1350327152489566
```

5. RESULT AND DISCUSSION

As you can see that mean absolute percentage error in the case of the regional model is less. Thus this implies that the smaller model is better fit for the region in comparison with the overall fitted model.

This could be very useful when we want to find predictions with precision and there is a sudden change in target. Ex. If there is a data set of housing and there are few houses which are extremely high priced in comparison to other houses of nearly the same size because of area or location.

But we need to be extra precious about overfitting. To avoided overfitting always remember to take sample size which is considerable.

Table 1: Model MAPE Measurements with key characteristics

Model Type	Mean Absolute Percentage Error (MAPE)	Key Characteristics
Overall Model	0.1439522	Trained on the entire dataset, struggles with curvilinear data, may overfit.
Small Regional Model	0.1350327	Trained on a sub-dataframe with high correlation to the target variable, better fit for localized data.

6. FUTURE SCOPE

1. Automated Split Point Determination is required which could make splitting easy in comparison to manual visual or mode based splitting.
2. This paper is only applied to Housing data set and it is required to be applied on different datasets.
3 This paper only split data into two parts and train one single small model, but it can be split into more small models to get better fit.
4. In this paper only one parameter is used that is absolute mean square error to compare, but one can use more and better parameters to compare.

7. CONCLUSION

In this paper, we introduced a novel approach to enhance the predictive accuracy of polynomial and linear regression models by implementing correlation-based dataframe splitting. By dividing the dataset into smaller, region-specific sub-dataframes based on the highest correlated feature with the target variable, we were able to create models that are more tailored to the local data patterns. Our experimental results demonstrate that these regional models outperform traditional global models, particularly in cases where there are sudden changes in the target variable, such as in housing prices influenced by location or other factors.

The proposed method effectively addresses the challenges of overfitting and poor fit in curvy or complex data by simplifying the model structure within each sub-region. However, careful consideration must be given to the selection of split points to avoid introducing bias or overfitting within the smaller data segments. Future work could explore automated methods for determining optimal split points and extending this approach to other types of models and datasets.

Overall, our findings suggest that correlation-based dataframe splitting is a promising technique for improving the accuracy and robustness of predictive models, particularly in scenarios where data exhibits regional variation.

REFERENCE

[1] Polynomial Regression: -https://saturncloud.io/blog/multivariate-polynomial-regression-with-python/

[2] Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition

[3] Andrew Ng’s course on ML specialization

[4] Andrew Ng’s Liberties for visualization: https://github.com/blaine12100/Andrew-NG-Machine-Learning-Course-in-Python/blob /master/Exercise3/utils.py

[5] Mitchell Wortsman. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time

[6] Lai, T. L., & Wei, C. Z,Least Squares Estimates in Stochastic Regression Models with Applications to Identification and Control of Dynamic Systems

[7] Weisberg, S. A tutorial on linear regression models

[8] Multiple Regression Analysis: Linear and Polynomial by Zhang, Z.

[9] Draper, N. R., & Smith, H, An Introduction to Polynomial Regression.

[10] Guyon, I., & Elisseeff, A, Improving Accuracy in Linear Regression with Feature Selection by