Amirkabir University of Technology
(Tehran Polytechnic)

Applied Machine Learning

Assignment IV

Alireza Nasoodi

401131059

Table of Contents

1- Why and how

1-1-1- Question:

How would you find the optimal number of random features to consider at each split?

1-1-2- Answer:

The optimal number of features to consider in a random forest model can vary based on the specific problem and dataset at hand. We can use these methods to find it:

a) **Grid Search**: Is a traditional way to perform hyperparameter optimization. It is searching through a manually specified subset of the hyperparameter space. In this case, we specify a range of plausible values for 'max_features' and then fit our model with each, choosing the one that performs best.

b) **Random Search**: This method doesn't test all possible values, but rather a fixed number of parameter settings is sampled from the specified distributions. This method is more efficient than grid search, especially when dealing with a large number of hyperparameters and their possible values.

1-2-1- Question:

What are the trade-offs between the different types of Classification Algorithms? How would do you choose the best one?

1-2-2- Answer:

Logistic Regression:

Strengths: Easy to implement, fast to train, provides probabilities for outcomes, and works well with linearly separable data.

Weaknesses: It assumes a linear decision boundary and is not flexible enough to capture complex patterns without feature engineering.

Decision Trees:

Strengths: Easy to interpret, handle both numerical and categorical data, and handle multi-output problems.

Weaknesses: Prone to overfitting and not as accurate as other methods.

Random Forest:

Strengths: High accuracy, handles overfitting better than Decision Trees, and doesn't require scaling of features.

Weaknesses: More computationally expensive and harder to interpret than Decision Trees.

Gradient Boosting:

Strengths: Often provides the highest accuracy in scenarios where the dataset is sufficiently large and complex.
Weaknesses: More computationally expensive and prone to overfitting if not properly tuned.

Naive Bayes:
Strengths: Simple, fast, and works well with high dimensional datasets.
Weaknesses: It assumes that all features are independent, which is not always the case in real-world scenarios.

K-Nearest Neighbors (KNN):
Strengths: Simple, works well with small datasets, and doesn't make any assumptions about the data.
Weaknesses: Computationally expensive with larger datasets and needs scaling of data.

Support Vector Machines (SVM):
Strengths: Effective in high dimensional spaces and can model non-linear decision boundaries.
Weaknesses: Computationally intensive with large datasets, requires scaling of data, and provides poor performance if the number of features exceeds the number of samples.

Choosing the best classification algorithm depends on specific task, the nature of data, and computational resources. Overall, we have to consider Number of features, Size of the dataset, Risk of overfitting or bias.

1-3-1- Question:

How does Ensemble Learning tackle the No-Free Lunch Dilemma?

1-3-2- Answer:

NFL says that there is no one model that works best for every problem. Ensemble learning helps to tackle the NFL dilemma in many ways like:

a) **Diversity**: Ensemble methods create a set of diverse models on a given problem. Each model may be best at handling a specific aspect of the problem, thus collectively, the ensemble covers a wider range of problem features than any individual model could.
b) **Error Reduction**: In ensemble learning, the predictions of several models are combined to produce the final output. This can significantly reduce errors in three ways:

   a. **Reducing bias**: If each individual model has high bias, an ensemble of these models may reduce the bias if the models are not all biased in the same way.
   b. **Reducing variance**: If each individual model has high variance, an ensemble can average out the variance and produce a more generalized model.
   c. **Reducing prediction noise**: If the error in the individual models is due to noisy data, then an ensemble can help reduce this by "averaging out" the noise.

c) **Better Generalization**: Due to their combined strength and reduced error, ensemble models often generalize better to unseen data. They can perform well across a variety of different problem sets, somewhat mitigating the NFL dilemma.

1-4-1- Question:

Can you use the LASSO method for Base Learner Selection?

1-4-2- Answer:

LASSO is a regression analysis method that performs both variable selection and regularization. It's capable of reducing the complexity of a model and preventing overfitting by imposing a constraint on the absolute sum of the coefficients. This can cause some coefficients to shrink to zero, effectively removing those variables from the model. We can use lasso in base learner selection, while doing so each base learner is considered a variable or feature. The LASSO method helps determine which base learners helps the most to the overall predictive performance by shrinking the coefficients of less significant base learners to zero. This effectively selects the most important base learners for the ensemble.

1-5-1- Question:

What's the similarities and differences between Bagging, Boosting, Stacking?

1-5-2- Answer:

Bagging creates multiple subsets from the original dataset, trains a classifier on each. All classifiers vote to classify new instances. It reduces variance and is less prone to overfitting. Random Forest is an example of bagging.

Boosting, like bagging, creates several classifiers, but it increases the weight of misclassified instances to give them more importance in subsequent models. It aims to reduce bias and, while robust to underfitting, can overfit if the underlying model is too complex. AdaBoost is an example.

Stacking, trains a "meta-learner" to make the final prediction based on the other classifiers predictions. Unlike bagging and boosting, stacking doesn't modify the

dataset or instance weights. It is flexible and can combine different types of models but is also more complex to implement and optimize.
And they all involve creating multiple classifiers and combining their predictions.

1-6-1- Question:

What is a dual and primal problem and how is it relevant to SVMs?

1-6-2- Answer:

The primal problem is the original optimization problem. It consists of minimizing or maximizing an objective function subject to constraints. In the SVMs, the primal problem involves finding the maximum-margin hyperplane that separates data into two classes.
The dual problem is a related optimization problem, derived from the primal problem, that provides a lower bound to the solution of the primal problem. The dual problem can be more efficient to solve than the primal problem, and it can also offer additional insights into the primal problem.
The dual formulation allows us to use the "kernel trick". This allows SVMs to model complex, non-linear decision boundaries. We can classify data that is not linearly separable in its original feature space by mapping it to a higher-dimensional space where it is linearly separable. Also, the dual formulation leads to a sparse solution, which means that only a subset of the input data points (the support vectors) are used to define the decision boundary. and the dual problem can be easier to solve numerically than the primal problem, especially when the input data is high-dimensional.

1-7-1- Question:

Can an SVM classifier outputs a confidence score when it classifies an instance? What about a probability?

1-7-2- Answer:

A SVM classifier can output a distance score, which can be interpreted as a measure of confidence in the classification decision. It does it by determining the distance from the instance to the decision boundary. Instances further from the decision boundary are classified with higher confidence, while those closer are classified with lower confidence.
SVM does not provide probabilities. However, it is possible to get probabilities by performing what's known as Platt Scaling on the output of the SVM. It means to fit a logistic regression model to the SVM scores to convert them into probabilities. This method is computationally expensive and adds an extra layer of complexity.

1-8-1- Question:

If you train an SVM classifier with an RBF kernel. It seems to underfit the training dataset: should you increase or decrease the hyper-parameter γ (gamma)? What about the C hyperparameter?

1-8-2- Answer:

γ defines how far the influence of a single training example reaches, with low values meaning 'far' and high values meaning 'close'. The model in this scenario is too biased or not complex enough. So, we would increase the gamma parameter. C is the penalty parameter of the error term. It controls the trade-off between smooth decision boundary and classifying the training points correctly. Small C creates a wider margin but allows more misclassifications. Large C creates a smaller margin because it wants to correctly classify as many instances as possible. in this scenario, we want to increase the value of C to allow the model to increase the penalty for misclassification and therefore fit the training data more accurately.

1-9-1- Question:

What technique can be used to solve the optimization problem cast by Support Vector Machines?

1-9-2- Answer:

SVMs optimization problem seeks to find the optimal hyperplane that maximizes the margin between two classes of data points in a multi-dimensional space. Sequential Minimal Optimization (SMO) breaks down the large Quadratic Programming (QP) problem generated by SVMs into a series of smaller problems that can be solved analytically, instead of using numerical optimization techniques.

1-10-1- Question:

How probabilities are calculated for SVM model?

1-10-2- Answer:

Platt Scaling is a commonly used method for obtaining probability estimates from SVMs. It fits a logistic regression model to the SVM's scores. Then these fitted logistic regression model provides the probability estimates.

1-11-1- Question:

In class we learnt that SVM can be used to classify linearly inseparable data by transforming it to a higher dimensional space with a kernel K(x, z) = ϕ(x) T ϕ(z), where ϕ(x) is a feature mapping. Let K1 and K2 be Rn × Rn kernels, K3 be a Rd × Rd kernel and c ∈ R+ be a positive constant. ϕ1 : Rn → Rd , ϕ2 : Rn → Rd , and ϕ3 : Rd → Rd are feature mappings of K1, K2 and K3 respectively. Explain how to use ϕ1 and ϕ2 to obtain the following kernels.

a. K(x, z) = cK1(x, z)

b. K(x, z) = K1(x, z)K2(x, z)

1-11-2- Answer:

(A) is simply a scaled version of the K1 kernel. Which do not necessarily need to use ϕ1 or ϕ2 to obtain this kernel. By multiplying the output of the K1 kernel by the constant c. we can reach to this state

In terms of the feature mapping ϕ1, the kernel can be expressed as:

$K(x, z) = \phi1(x)T\ \phi1(z) * c$

That is, compute the inner product of ϕ1(x) and ϕ1(z) first and then multiply the result by c.

(B) If ϕ1(x) is a feature mapping for K1, and ϕ2(x) is a feature mapping for K2, then we can define a new feature mapping ϕ(x) as below:

$\phi(x) = \phi1(x) \odot \phi2(x)$,

where $\odot$ denotes the component-wise product.

The resulting kernel K is then defined by:

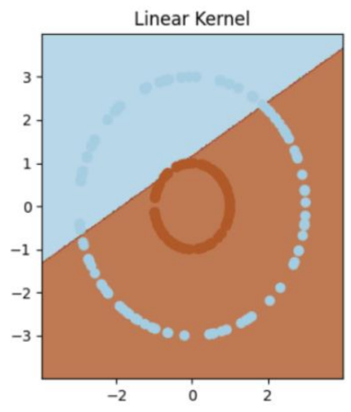
$K(x, z) = \phi(x)T\phi(z) = (\phi1(x) \odot \phi2(x))T(\phi1(z) \odot \phi2(z)) = K1(x, z)K2(x, z)$.

1-12-1- Question:

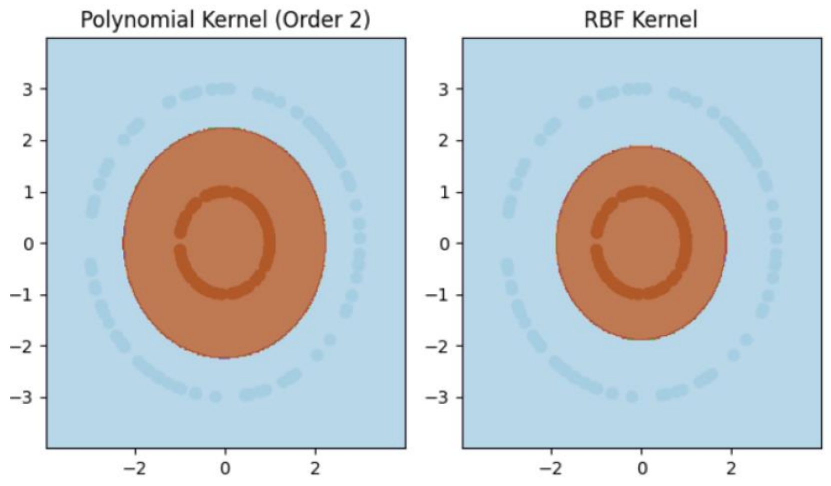
You are given the following 3 plots, which illustrates a dataset with two classes. Draw the decision boundary when you train an SVM classifier with linear, polynomial (order 2) and RBF kernels respectively. Classes have equal number of instances.

1-12-2- Answer:

In these three pictures we have a two type of data with different radius but with same center. As we can geuss linear distinguish can not act properly as we cant separate these datas with only a line.



Both Poly and rbf kernel can distinguish the data easily as the try to differentiate the data in higher dimension.



2- Custom Dataset

In this question we use ‘make_classification’ from sklearn and make a dataset with 1000 datas and 5 features which 3 of them are informative. Then we split our data set into train and test with ratio of 80% to 20% then we use standard scaler inorder to scale our data. For our grid serach we try 4 different c’s, 3 different kernels and 3 different gamma value to find the best model possible with these combinations. After finding the best model which after running the program would be {'C': 10, 'gamma': 0.1, 'kernel': 'rbf'} we fit the test data based on the best model and then we calculate the Precision and Recall of the model which with the model we trained would be {Precision: 0.95 , Recall: 0.96}.

MKL stands for Multiple Kernel Learning. MKL allows for the use of multiple kernels instead of a single kernel. Each kernel captures different aspects of the dataframe, so the combination of these kernels could result in a more accurate model. After implementing this method we can calculate Precision and Recall of the model as {Precision: 0.95 , Recall: 0.96 }.

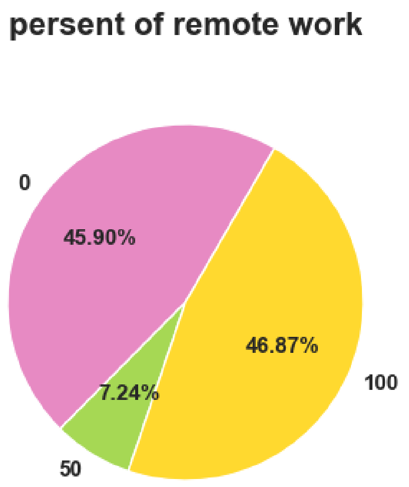
3- Credit Card Fraud

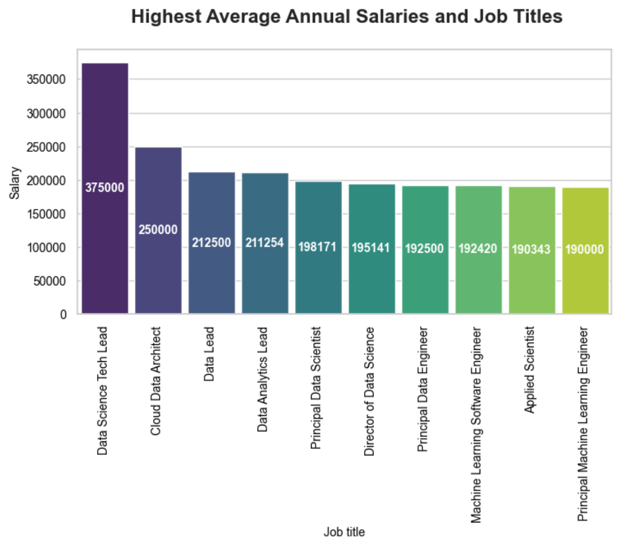
This dataset has unbalanced data problem that if we use the data as it is it would cause undersampling problem. We split the fraud and non-fraud data and appropriate to fraud data, a sample of non-fraud data is selected so that the training data has an almost equal ratio of fraud and non-fraud data and the training model does not perform badly.

After splitting our dataset first we scale our data into train and test then we scale our data with standard scaler then we try to train our model with oneClass svm and we used grid search in order to find the best parameters for this model which after evaluating it the best model for it would be . It would give us these metrics {Precision: 0.0010152284263959391 Recall: 0.06666666666666667 F1 Score: 0.002}

4- Data Science Salaries

After analyzing the data we can see that columns salary and salary_currency are not helpful to our model so we drop them. We visualize the data as:

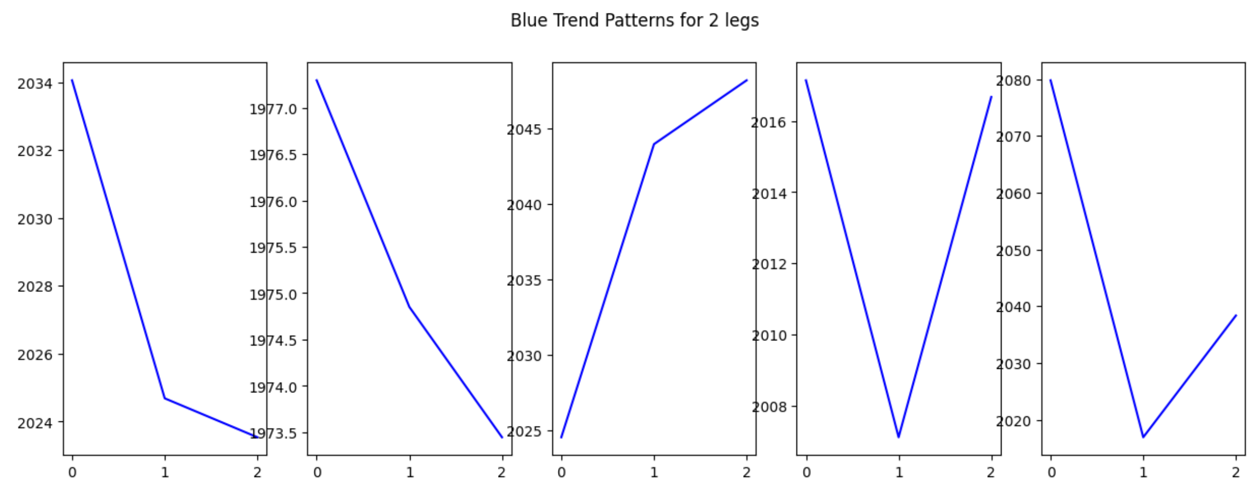


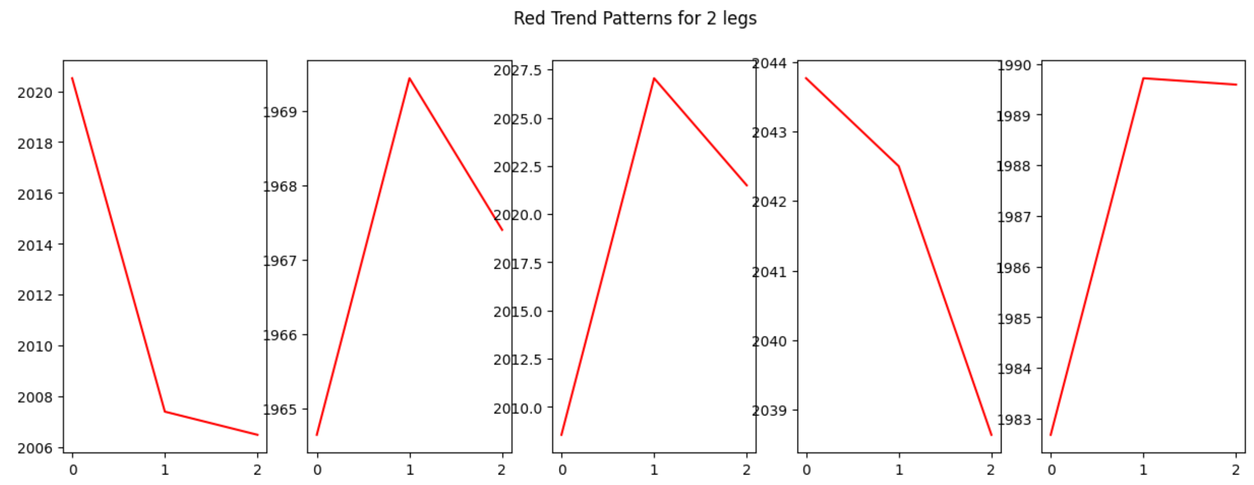


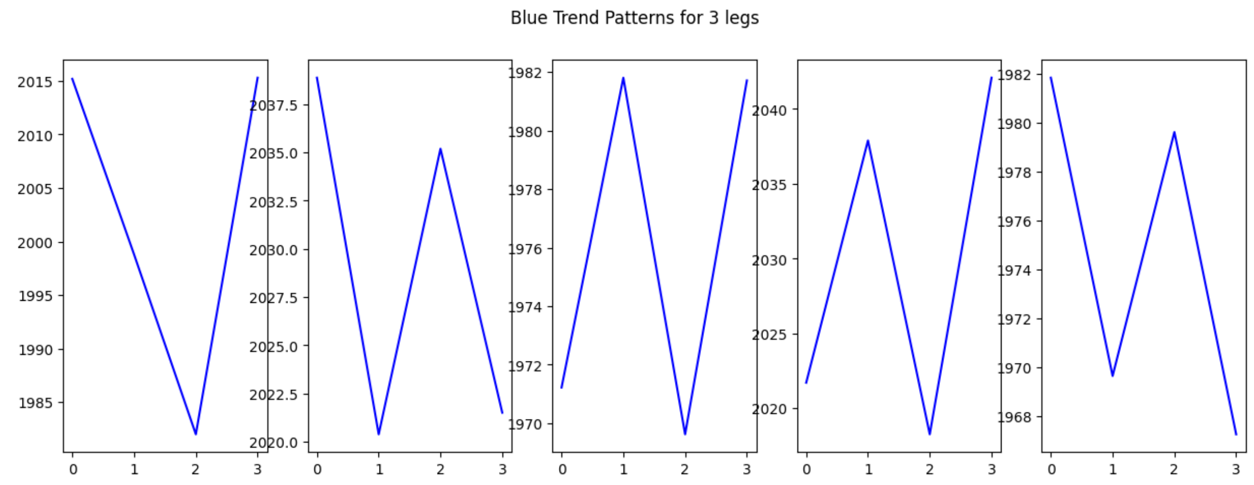
We delete the duplicated data in our data set and categorize our data after doing so now we can try to train our model using decision tree with depth 2 which its accuracy is 0.031 we extract the leaf node assignments for the training set using the 'apply' method of the decision tree. Then, we train a Random Forest model using the leaf node assignments after applying it we can see that the accuracy of the model is slightly increased to 0.33.

5- Price Pattern Extraction

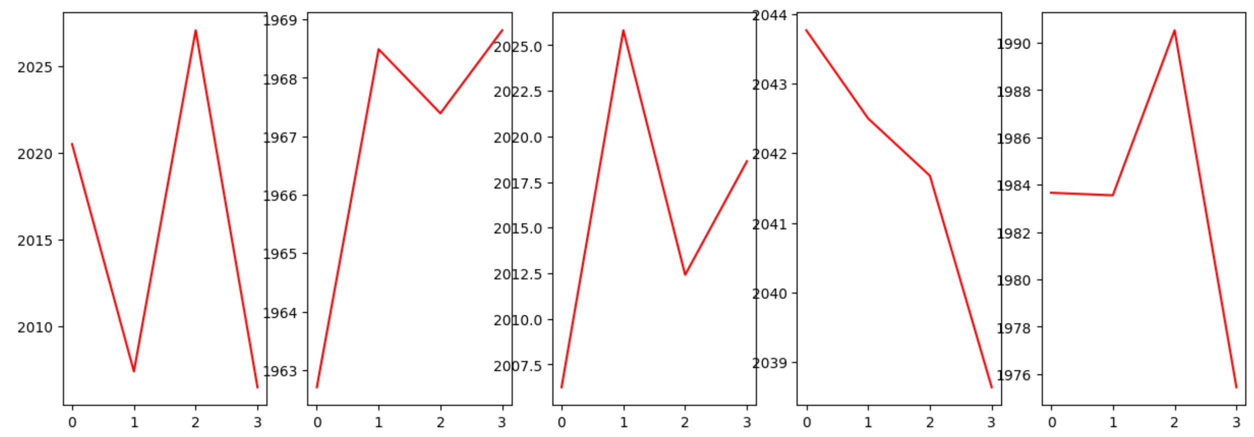
We use start price and end price and also the pattern of the line as our needed features for developing KMeans. We buil three datasets with 2,3 and 4 legs and for every one of them we train a KMeans model which every one of them will have a pattern for each of the colors in our dataset 'blue' and 'red' we developed a way to show the pattern of every cluster with number of legs in every dataset.


Blue Trend Patterns for 2 legs

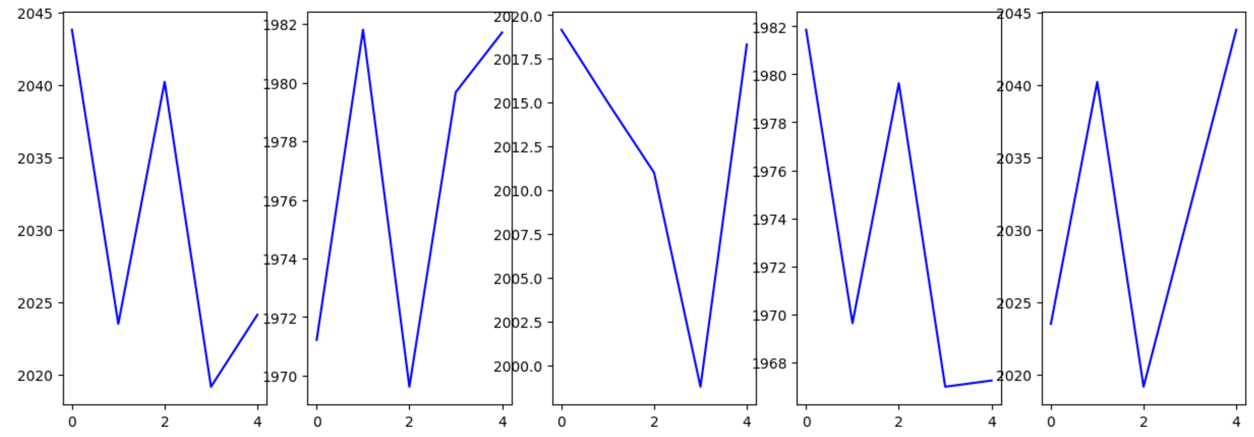

Red Trend Patterns for 2 legs


Blue Trend Patterns for 3 legs

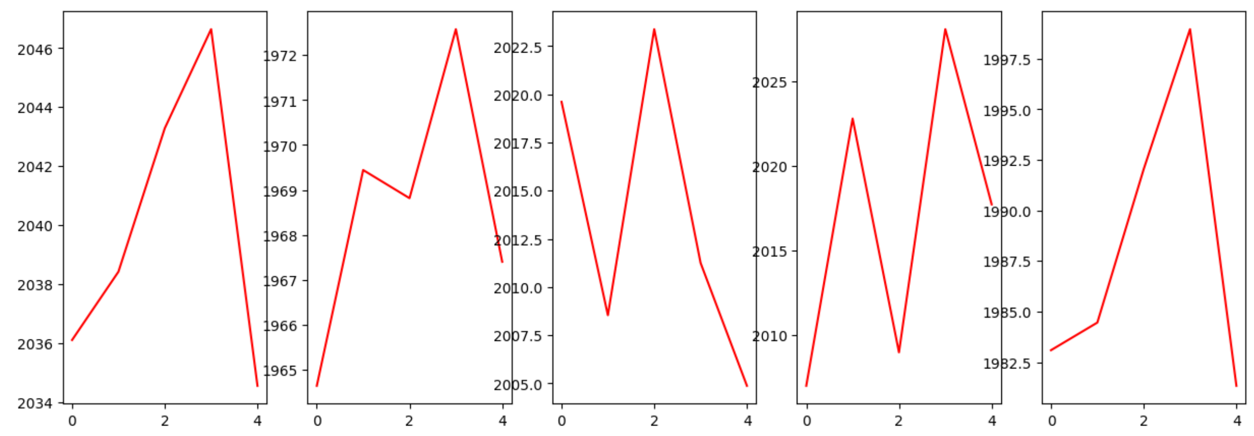
Red Trend Patterns for 3 legs



Blue Trend Patterns for 4 legs

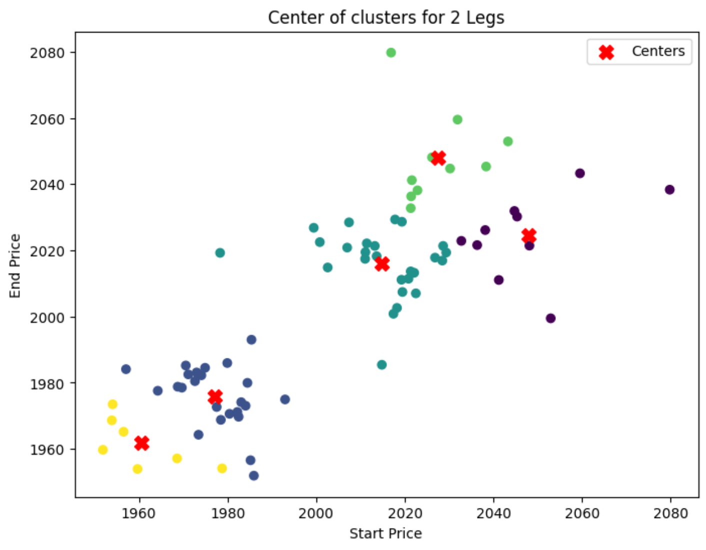


Red Trend Patterns for 4 legs

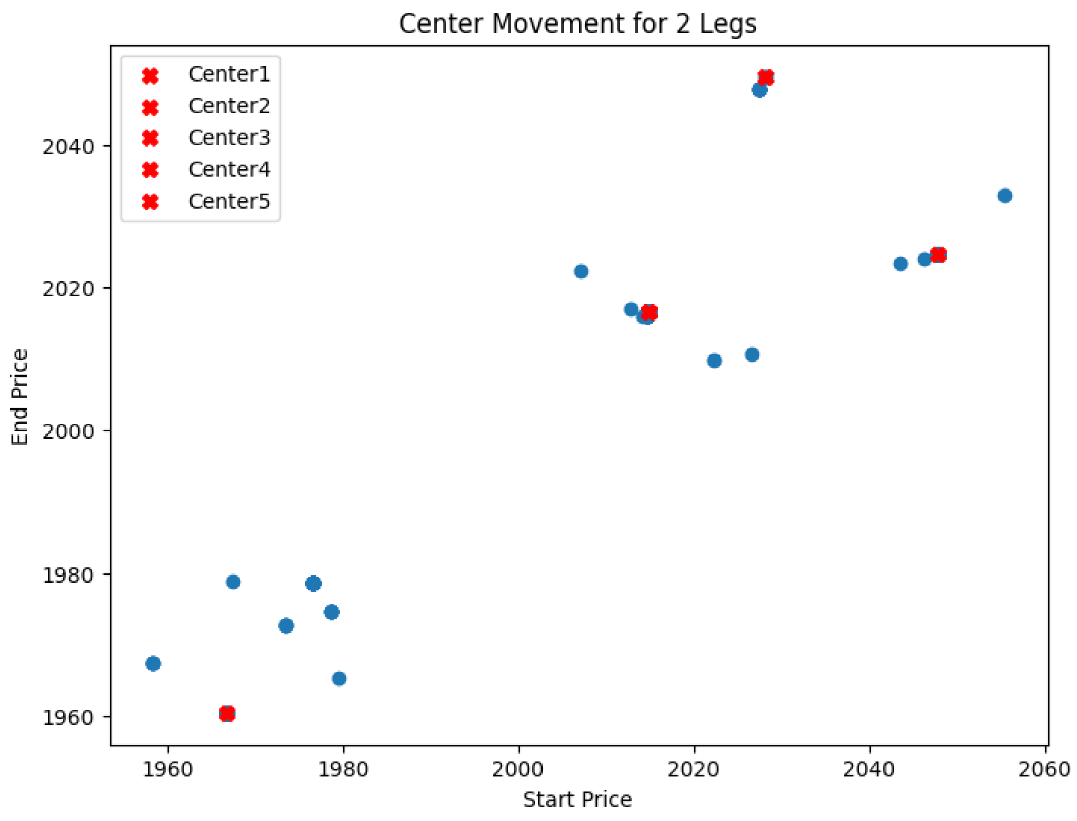


As you can see in these pictures we have developed trained KMeans on each trend and extracted the behavior of the pattern.

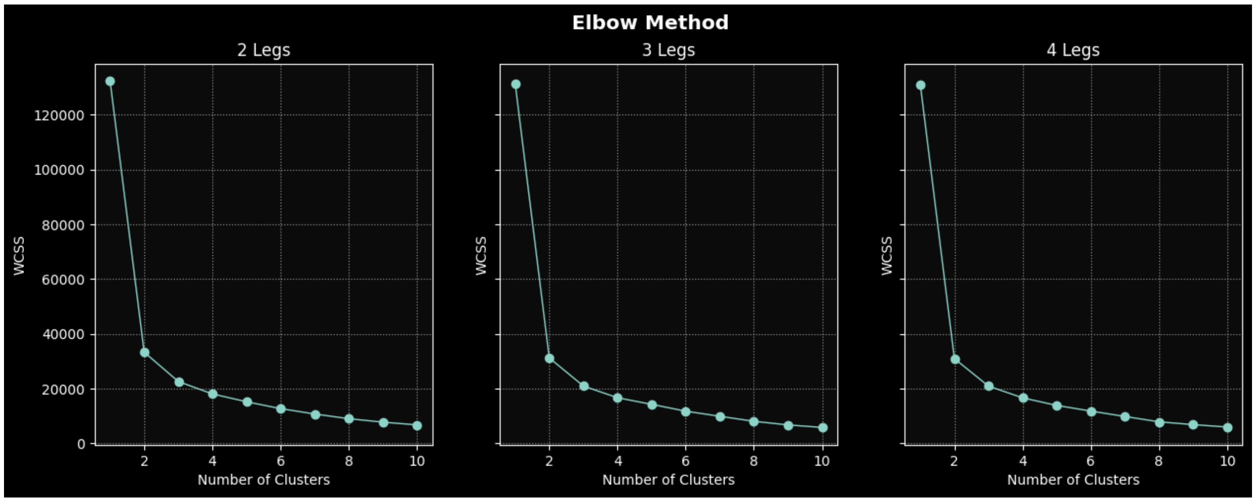
We also extracted the centers of our dataset with two legs



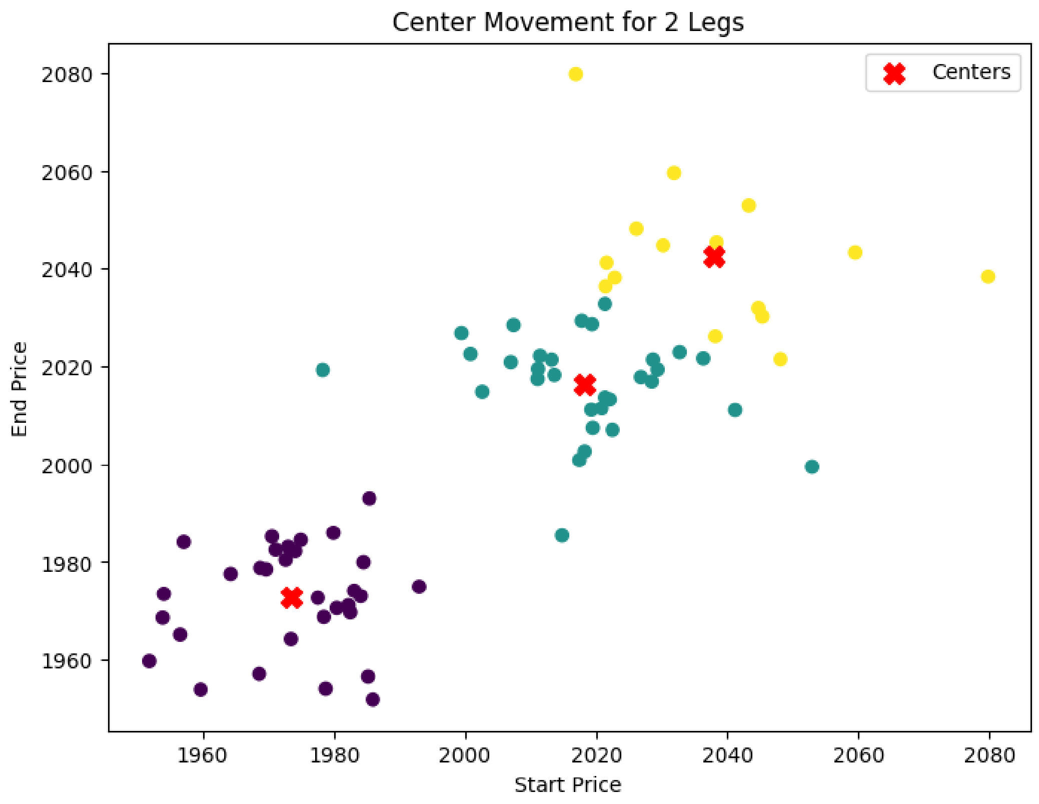
We also developed a method to identify the movement of the centers of clusters thet you can see it below:



For maximizing our performance we developed an elbow chart in order to find the best hyperparameter requerd for KMeans:

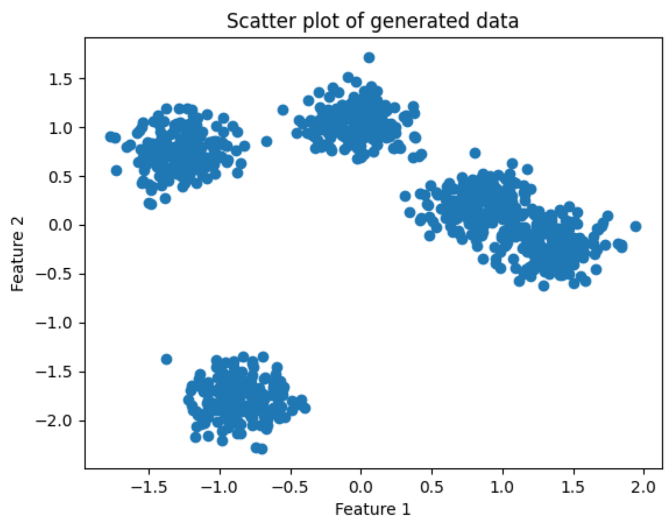


As you can see KMeans with 3 clusters would work better we can see that in the chart below:



6- Synthetic Clusters

We use ‘make_blobs’ in order to generate synthetic clusters and we use standard scaler to scale our data. We plot this dataset using matplot. Then we use dbscan to train our model.



We set eps to 0.1 and minimum samples required to make a cluster to 5. After trainig the model the evaluation of the model is measured and they are:

Homogeneity: 0.7884160194686332

Completeness: 0.8587051088745988

V-measure: 0.822060809234197

Adjusted Rand Index: 0.7250293466450323

Adjusted Mutual Information: 0.8208376480983665

Silhouette Coefficient: 0.5775766408408879