Table of Contents

1. INFORMATION OF DATASETS

For clustering and feature selection, 2019 World Happiness Report dataset used which is a survey of the state of global happiness taken from the Gallup World Poll. In total we have 8 features to compare. Country is the name of the countries attended the survey. Happiness score is the final scores that estimate the extent of six happiness factors below. GDP per capita is a measure of a country's economic output that accounts for its number of people. Social support means having friends and other people, including family, to turn to in times of need or crisis to give you a broader focus and positive self-image. Healthy Life Expectancy is the average number of years that a newborn can expect to live in "full health"—in other words, not hampered by disabling illnesses or injuries. Freedom of choice describes an individual's opportunity and autonomy to perform an action selected from at least two available options, unconstrained by external parties. Generosity is the quality of being kind and generous. A perception of corruption is according to The Corruption Perceptions Index (CPI) is an index published annually by Transparency International since 1995 which ranks countries "by their perceived levels of public sector corruption, as determined by expert assessments and opinion surveys.

For PCA, mushrooms dataset is used. This dataset includes descriptions of hypothetical samples corresponding to 23 species of gilled mushrooms in the Agaricus and Lepiota Family Mushroom drawn from The Audubon Society Field Guide to North American Mushrooms (1981).

The COVID-19 dataset contains results of a survey about emotional responses to COVID-19. 2491 participants were asked to indicate their emotions and express these in numerical scores. For logistic regression, decision tree and random forest, Covid – 19 data is used. Due to the page constraint of project report logistic regression and random forest were not included in this report. The related python file contains random forest and logistic regression.

2. SUBSET SELECTION FOR LINEAR REGRESSION

For subset selection Happyness Data set is used. The independent variables in the model analyzed is shown below. Whole processes for Subset Selection and Shrinkage Method (it will be shown in the next sections) will function on these variables.

One of the most important parts of subset selection for linear regression is to divide dataset into training and test sets with chosen test size, which is 0.2 for this research.

2.1. BEST SUBSET SELECTION

In Best Subset Selection approach, features must be selected which gave the best model precisely. Best MSE result is 0.33 with Best Subset Selection with subset (0, 1, 2, 3, 5) which is provided by 57^{th} subset as seen below when subset average scores sorted from highest to lowest which results as MSE1 = 18.57%.

	feature_idx	cv_scores	avg_score	feature_names	ci_bound	std_dev	std_err
57	(0, 1, 2, 3, 5)	[-0.25978957518240464, -0.27968626896873094, -...	-0.333475	(GDP per capita, Social support, Healthy life ...	0.0751715	0.058486	0.029243
41	(0, 1, 2, 3)	[-0.2569198882919649, -0.3095091597495441, -0....	-0.33419	(GDP per capita, Social support, Healthy life ...	0.0699185	0.0543989	0.0271995
56	(0, 1, 2, 3, 4)	[-0.2466065463436417, -0.30770393398149387, -0...	-0.337527	(GDP per capita, Social support, Healthy life ...	0.0798159	0.0620995	0.0310497
62	(0, 1, 2, 3, 4, 5)	[-0.25659343575048554, -0.28313549000026783, -...	-0.33891	(GDP per capita, Social support, Healthy life ...	0.0826379	0.0642951	0.0321476
45	(0, 1, 3, 5)	[-0.2751610787243343, -0.2614355774063514, -0....	-0.347146	(GDP per capita, Social support, Freedom to ma...	0.0899722	0.0700014	0.0350007
...	...	...	...	...	...	...	...
3	(3,)	[-0.5915619941744733, -0.8115701671432194, -1....	-0.877765	(Freedom to make life choices,)	0.245376	0.190911	0.0954555
18	(3, 4)	[-0.6407871476737464, -0.7725271184731561, -1....	-0.894454	(Freedom to make life choices, Generosity)	0.209913	0.16332	0.0816599
5	(5,)	[-0.9739154029066242, -0.8482271644357512, -1....	-1.1179	(Perceptions of corruption,)	0.27517	0.214092	0.107046
20	(4, 5)	[-0.9713127015481244, -0.837833578657559, -1.4...	-1.13264	(Generosity, Perceptions of corruption)	0.298074	0.231912	0.115956
4	(4,)	[-1.0337602587147425, -1.2584048758414599, -1....	-1.30065	(Generosity,)	0.284782	0.22157	0.110785

```
linear_model = LinearRegression()
x_train_selected = x_train.iloc[:,[0, 1, 2, 3, 5]]
x_test_selected = x_test.iloc[:,[0, 1, 2, 3, 5]]

linear_model.fit(x_train_selected, y_train)

pred0 = linear_model.predict(x_test_selected)
mse1 = mean_squared_error(y_test, pred0)
print("MSE1 (Best Subset Selection MSE):", format(mse1, ".4f"))
```

```
MSE1 (Best Subset Selection MSE): 0.1857
```

2.2. STEPWISE SELECTION

2.2.1. FORWARD SELECTION

Forward stepwise selection approach is starting with a null model including no independent variables, and then adds independent variables to the model. In Happines Data Model, there are 6 predictors so the functions setted going from start to **k_features = 6. "forward"** option was selected **True**. In this approach mean squared error used but in negative so the values can be easily sorted from highest to lowest.

```
from mlxtend.feature_selection import SequentialFeatureSelector as SFS
linear_model_2 = LinearRegression()

from mlxtend.feature_selection import SequentialFeatureSelector as SFS
linear_model_2 = LinearRegression()

sfs = SFS(linear_model_2, k_features=6, forward = True, floating = False, scoring = 'neg_mean_squared_error', cv=5)

feature_names = ("GDP per capita", "Social support", "Healthy life expectancy", "Freedom to make life choices",
                 "Generosity", "Perceptions of corruption")
sfs = sfs.fit(x_train, y_train, custom_feature_names = feature_names)

sfs.subsets_
```

As likely to Best Subset Selection, the optimal subset is identifying as (0, 1, 2, 3, 5). Hence, for training set, 0, 1, 2, 3, 5th indexed features are selected since they will give the optimal solution.

	feature_idx	cv_scores	avg_score	feature_names	ci_bound	std_dev	std_err
5	(0, 1, 2, 3, 5)	[-0.25978957518240464, -0.27968626896873094, -...	-0.333475	(GDP per capita, Social support, Healthy life ...	0.0751715	0.058486	0.029243
4	(0, 1, 2, 3)	[-0.2569198882919649, -0.3095091597495441, -0....	-0.33419	(GDP per capita, Social support, Healthy life ...	0.0699185	0.0543989	0.0271995
6	(0, 1, 2, 3, 4, 5)	[-0.25659343575048554, -0.28313549000026783, -...	-0.33891	(GDP per capita, Social support, Healthy life ...	0.0826379	0.0642951	0.0321476

```
# After fitting training and test sets, MSE of the Forward Selection process has shown below.

from sklearn.metrics import mean_squared_error
from sklearn.linear_model import LinearRegression
linear_model1 = LinearRegression()
x_train_selected1= x_train.iloc[:,[0, 1, 2, 3, 5]]
x_test_selected1= x_test.iloc[:,[0, 1, 2, 3, 5]]
lm = linear_model.fit(x_train_selected1,y_train)

pred2 = lm.predict(x_test_selected1)
mse2=mean_squared_error(y_test,pred2)
print("Forward Selection MSE: ", format(mse2,".4f"))
```

```
Forward Selection MSE:  0.1857
```

After the score of subset calculated which includes 0, 1, 2, 3, 5. predictors and it results in **MSE2 = 18.57%.** This will be the first Mean Squared Error value came from methods will be used for this dataset.

2.2.2. BACKWARD SELECTION

In opposite of forward selection, backward selection runs by starting with the full model including all p (6 for current dataset) predictors, and then iteratively removes the least useful predictor, one-at-a time.

"k_features" is also identified as 6 here since number of predictors does not change. But in Backward Selection, “forward” option must be identified as False because this functions by excluding predictors from the model. As predicted, the MSE score for Backward Selection is also 18.57% which is the same as in Forward Selection.

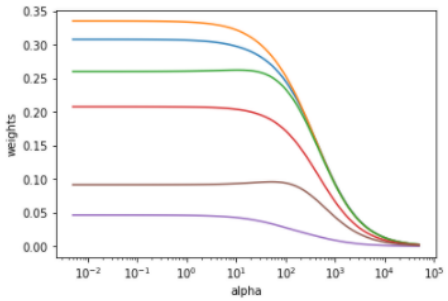
2.3. SHRINKAGE METHODS

2.3.1. RIDGE REGRESSION

In Ridge Regression Method, selecting a good value for λ is the most critical step and cross validation is being applied to obtain λ. So, one of the priorities is to standardizing variables after alphas were created. For each alpha value, “Ridge Regression” models fitted.

```
#To examine the model, visualization of the coefficients of the different alfas was proceeded.
ax = plt.gca()
ax.plot(alphas, coefs)
ax.set_xscale('log')
plt.axis('tight')
plt.xlabel('alpha')
plt.ylabel('weights')
```

```
Text(0, 0.5, 'weights')
```



By applying cross validation to lambdas, lambdas were sorted according to their validation errors.

```
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
lambdas = np.linspace(0.01,1000,num=10000)
scoresCV = []
for l in lambdas:
    ridge_reg = Ridge(alpha=l)
    ridge_reg.fit(x_train, y_train)
    scoreCV = cross_val_score(ridge_reg, x_train, y_train, scoring='neg_mean_squared_error',
                              cv=KFold(n_splits=10, shuffle=True,
                                       random_state=1))
    scoresCV.append([l,-1*np.mean(scoreCV)])
df2 = pd.DataFrame(scoresCV,columns=['Lambda','Validation Error'])
df2
# Lambdas and Validation Errors for each index is shown as a table below
```

	Lambda	Validation Error
0	0.010000	0.349669
1	0.110009	0.346350
2	0.210018	0.343919
3	0.310027	0.342128
4	0.410036	0.340815
...	...	...
9995	999.599964	1.257343
9996	999.699973	1.257348
9997	999.799982	1.257354
9998	999.899991	1.257359

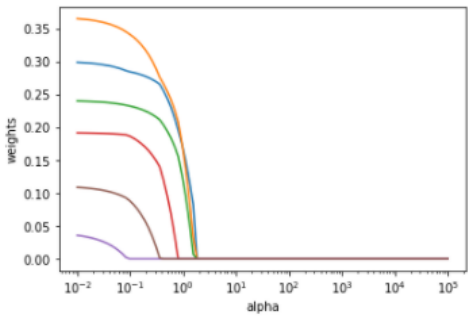
For next step, Cross Validation is selected as 10, negative mean squared error method is chosen for scoring and with training y and x sets, best alpha value is obtained. Best Alpha found 0.6610. Further, Ridge was fitted by using training and calculating mean square error by using test set and MSE for Ridge Regression is obtained. MSE4 found 18.64 %.

Final Coefficients for Ridge Regression are GDP per capita, social support , healthy life expectancy , freedom to make life choices, generosity and corruption

2.3.2. THE LASSO

```
lasso = Lasso(max_iter = 10000)
coefs = []

for a in alphas:
    lasso.set_params(alpha=a)
    lasso.fit(scale(x_train), y_train)
    coefs.append(lasso.coef_)

ax = plt.gca()
ax.plot(alphas*2, coefs)
ax.set_xscale('log')
plt.axis('tight')
plt.xlabel('alpha')
plt.ylabel('weights')
```

```
Text(0, 0.5, 'weights')
```



Again, Cross Validation is selected as 10, negative mean squared error method is chosen for scoring. However, Lasso aims to create models with less predictor in general. After obtaining the validation errors of lambdas, Lasso alpha value found as 0.00177. The MSE5 for Lasso Model must be calculated by test sets of predictors and y, found 18.35%.

```
lassocv = LassoCV(alphas = None, cv = 10, max_iter = 100000)
lassocv.fit(x_train, y_train)
lasso.set_params(alpha=lassocv.alpha_)
lasso.fit(x_train, y_train)
print("Lasso Alpha: ", lassocv.alpha_)
```

```
Lasso Alpha:  0.001778055592111914
```

```
#In this part, we aim to find the mean squared error in lasso process
mse5 = mean_squared_error(y_test, lasso.predict(x_test))
print("The Lasso MSE:", format(mse5, ".4f"))
```

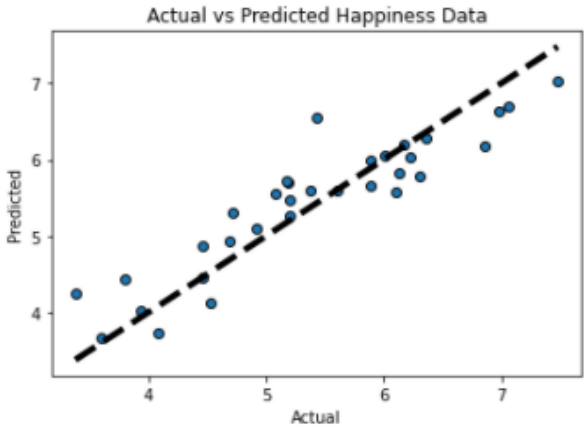
```
The Lasso MSE: 0.1835
```

2.4. RESULTS AND MODEL SELECTION

Mean Squared Errors for Methods:

- Best Subset Selection MSE1: 0.1857
- Forward Selection MSE2: 0.1857
- Backward Selection MSE3: 0.1857
- Ridge Regression MSE4: 0.1864
- The Lasso MSE5: 0.1835

As seen above, MSE5 is the smallest, The Lasso model is chosen.



With Lasso Model, the graph above is obtained. And it can be seen the actual and predicted values are corresponds with each other that means it has a low error with Lasso Model. Adding new the features with high cross validation scores iteratively or removing the features from the full model did not give the minimum error neither.

So, it can be said that Best Subset Selection Methods; Forward and Backward Selections were not providing the optimal model since their Mean Squared Error was 18.57%.

In Ridge Regression models, model standardized coefficients at the beginning and than worked on them. Whenever Ridge Regression is applied, it is better to divide it into its standart deviations. Because if they are not in the same scale, reducing them might have troubles. We picked the best alpha using validation on the training set. Application the Ridge Regression gave the highest MSE in our dataset which is 18.64%.

In Lasso models, penalty has the effect of forcing some of the coefficient estimates to be exactly zero by using lambda value. In Ridge the feasible region is circle whether in Lasso usually is corner point. In lasso, as isocountour line gets bigger, SSE will increase.

As seen above, all the 6 features in our model which are: GDP per capita, Social support, Healthy life expectancy, Freedom to make life choices, Generosity, Perceptions of corruption is used in Lasso Method. None of the features had the 0.0000 coefficient so, it was a must to use all of them.

3. DECISION TREE METHOD

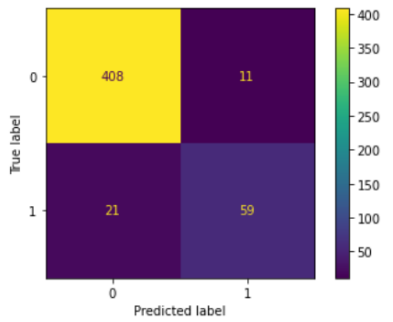
For decision tree, Covid 19 Data Set is used for searching best parameters which are number of trees and number of features.

The values for Max_depth [3, 4, 5, 6, 7, 8, 9, 10] and ccp_alpha [0, 0.02, 0.05, 0.08, 0.1] values were evaluated and the best results observed as [4] for max_depth and [0] for ccp_alpha.

The rmse value of the train set observed as 0.226 and the rmse value of the test set observed as 0.275.

Confusion Matrix:

```
# Confusion matrix plotted.

plot_confusion_matrix(model, X_test, y_test)
plt.show()
```



According to the confusion matrix accuracy is 93,59%. True positive rate is 73,8%.

4. CLUSTERING

For clustering, Happiness Data Set is used again for grouping data in order to similarities. As clustering methods; K-means clustering, hierarchical clustering and density based clustering techniques applied in Python. For page limits, only K-means clustering part explained.

Before using any metric, features are scaled using MinMaxScaler(), which brings down the values between 0 and 1. Then, Country column is set as an index column.

```
# Feature Scaling using MinMaxScaler().

scaler = MinMaxScaler()

scaled_data = scaler.fit_transform(X)

df_scaled = pd.DataFrame(scaled_data, columns = X.columns, index = df["Country"])
df_scaled
```

4.1. K-MEANS CLUSTERING

In this part a list is created for K values between 2 and maximum number of features, 6. Then, empty lists created for checking both distortion and silhouette scores.

After the kmeans.fit(df_scaled), k-means automatically calculates the distortions. For silhouette scores, silhouette_score funstion takes two arguments. One of them is original dataframe and the second one is its cluster labels.

KMeans function takes n_init parameter 10 by default. It chosen 20 which means KMeans algorithm tries 20 different starting points, and chooses the best point for starting.

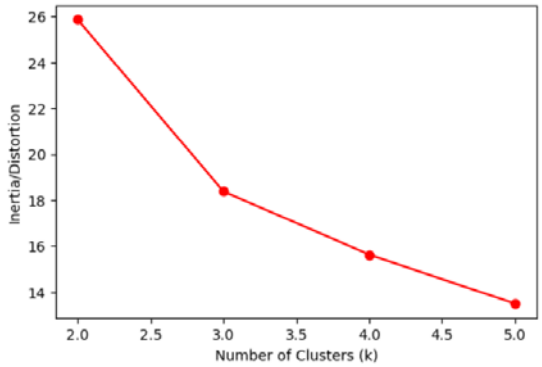
K-Means++ tries to place initial cluster centroids as far as possible which used by simply adding KMeans function to init=k-means++ parameter.

```
# Clustering with K-means algorithm.

kList = np.arange(2,6)

distortionList =[]
silhouetteList = []

for k in kList:
    kmeans = KMeans(n_clusters = k, n_init = 20, init='k-means++')
    kmeans.fit(df_scaled)
    distortionList.append(kmeans.inertia_)
    silhouetteList.append(silhouette_score(df_scaled , kmeans.labels_))
```

4.2. ELBOW METHOD

One way to find the optimal number of clusters is elbow method. This method tries different K values and records the distortions for each K values.

```
# Elbow Method for finding best k.

plt.figure(dpi=100)
plt.plot(kList , distortionList, marker="o",color='r')
plt.xlabel("Number of Clusters (k)")
plt.ylabel("Inertia/Distortion")
plt.show()
```

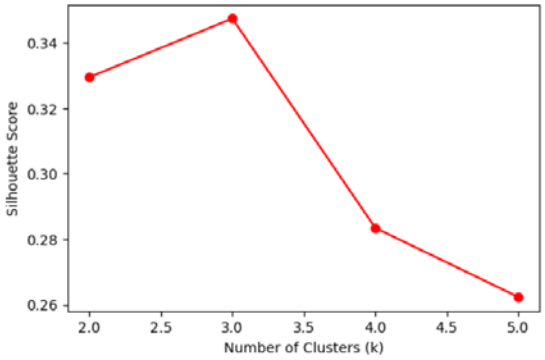


As seen above; the point where the reduction of distortion lowest the most, such an elbow is identified when K = 3, it can be candidate for optimal number of clusters.

4.3.SILHOUETTE COEFFICIENTS METHOD

In this method, the mean of the scores of all data points are calculated. Silhouette list is plotted to see the results.

```
# Silhouette Scores for finding best k.

plt.figure(dpi=100)
plt.plot(kList , silhouetteList, marker="o",color='r')
plt.xlabel("Number of Clusters (k)")
plt.ylabel("Silhouette Score")
plt.show()
```



As seen above, K = 3 gives the biggest silhouette score which gives the same K result with elbow method. So, K = 3 chosen as optimal clusters for this model.

4.4.FITTING MODEL WITH THE OPTIMAL CLUSTERS

With chosen K = 3 from the methods above, model fitted again and results interpreted.

```
# Model fitted again with K=3.

kmeans = KMeans(n_clusters = 3, n_init = 15)
kmeans.fit(df_scaled)
```

```
# Interpretation of Final Clusters

kmeans.cluster_centers_
results = pd.DataFrame(kmeans.cluster_centers_ , columns = X.columns)
results
```

	GDP	Social support	Healthy life expectancy	Choice freedom	Generosity	Perceptions of corruption
0	0.619939	0.808302	0.714306	0.616230	0.257368	0.155855
1	0.241131	0.537227	0.371699	0.512506	0.383123	0.225645
2	0.827413	0.921574	0.872719	0.875018	0.485223	0.636063

According to results;

- Social support value is close to 1 in cluster 2.

- Healthy life expectancy value is close to 1 in 2 cluster as well.

- In 2. cluster, most of the countries in these clusters have higher life expectancy.

- On average, countries in the 2. cluster has more corruptions.

- Generosity of the 0. cluster is lower than others, so it can be said that these countries people are less kind and generous.

5. DIMENSIONALITY REDUCTION WITH PRINCIPAL COMPONENT ANALYSIS

For PCA, Mushrooms Data Set is used for combining features and producing synthetic features that are linear combinations of the original features. Both manual approach of PCA and scikit-learn approach of PCA is coded in the py. file. The PCA class is scikit-learn's class. Model fitted using training set, than transformed training and test sets scaled before. In the end, classified using logistic regression.

When number of components is equal to 12, it gives the higher accuracy score.

```
# Instead of applying all the steps above, the PCA can be used.

pca = PCA(n_components=12)

X_train_pca = pca.fit_transform(X_Train_scaled)
X_test_pca = pca.transform(X_Test_scaled)

lr = LogisticRegression()
lr.fit(X_train_pca, y_train)
predictions = lr.predict(X_test_pca)
print(classification_report(y_test, predictions))
```

```
              precision    recall  f1-score   support

           0       0.89      0.95      0.92      1263
           1       0.94      0.87      0.91      1175

    accuracy                           0.92      2438
   macro avg       0.92      0.91      0.91      2438
weighted avg       0.92      0.92      0.91      2438
```

According to the results;

- Model has %92 accuracy, which is good.

- According to f1-scores, model did a very good job classifying class 0 and class 1.

- So, n_components should be chosen as 12.

6. CONCLUSION AND COMMENTS

Happiness Scores were different in different countries and there are many factors may affect this response. Most significant predictors in these scores were selected as GDP per capita, Social support, Healthy life expectancy, Freedom to make life choices, Generosity, Perceptions of corruption. After obtaining this data, different techniques have been used to obtain which predictors may have major effect on the response variable which is Happiness Scores in Countries.

After trying 5 different techniques and calculating Mean Squared Errors for each of them, the minimum error is resulted by Lasso Model. Hence, we had calculated the coefficients of predictors in this model to decide which may not have a significant effect or not. However, it is found out that each 6 factors/ predictors in Happiness Scores must be assumed significant. None of them cannot be ignored or removed in any processes. Generally, it is common to see

that Lasso eliminates some of the features to obtain a smaller MSE, yet, in this example, clearly all features were important for response variable.