Part 1: Classification problem

The logistic regression and linear discriminant analysis (LDA) models provide comparable predictive performance for smoking status (Smoker) as the dependent variable.
The logistic regression analysis reveals the effects of each predictor on the log-odds of being a smoker. A high coefficient suggests that the likelihood of being a smoker decreases as the predictor increases. The second model, which includes squared and interaction terms, can capture non-linear relationships and interactions between predictors, potentially improving the predictive performance.

The mean cross-validation scores for both models (~ 0.600 for the first and ~ 0.603 for the second) indicate a moderate ability to predict smoking status. The slight increase in the second model's score suggests that including non-linear terms may enhance the model's predictive performance.

Logistic regression

```
# Define the variables
predictors1 = ['Income', 'age', 'PCIGS79']
x1= logit1[predictors1]
y = logit1['Smoker']

# Estimate the model
logit1reg = LogisticRegression()
lr=logit1reg.fit(x1, y)
print(lr.coef_)

# Let’s cross validate the results of the logit regression above using k-10
validation
scoreslr = cross_val_score(lr, x1, y, cv=10)
print(scoreslr)

# Let’s find the average of the above logit scores
scoreslr.mean()

Mean score is: 0.600329131652661
```

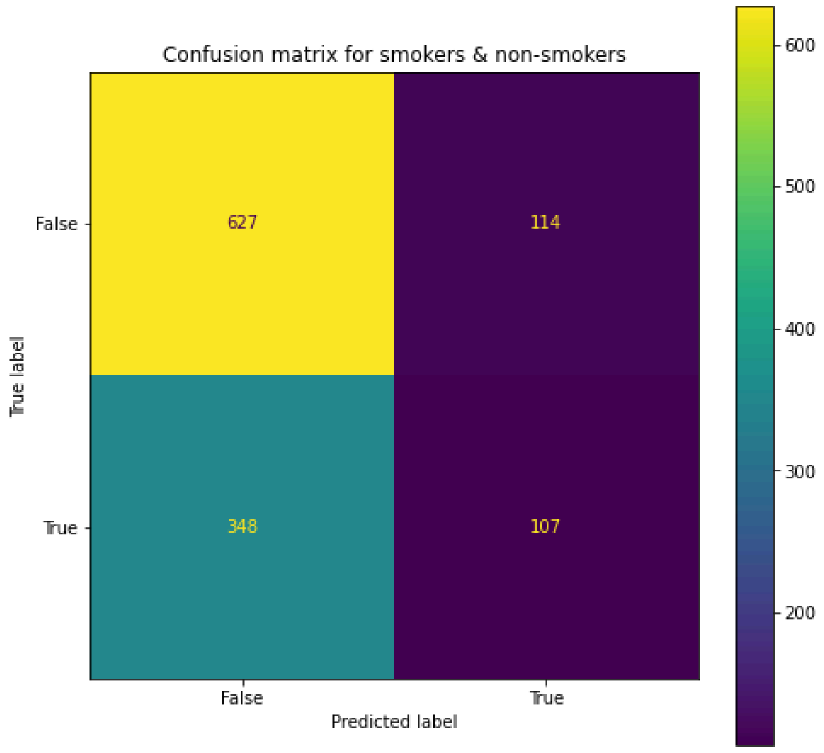
Adding Interaction terms and squared terms

```
# Square dependent variable:
logit1['Incomesq'] = logit1['Income']**2
logit1['agesq'] = logit1['age']**2
logit1['PCIGS79sq'] = logit1['PCIGS79']**2

# Interaction-terms:
logit1['IncomeagePCIGS79'] =
logit1['Income']*logit1['age']*logit1['PCIGS79']

# Estimate the model with square and non linear terms
   # Define predictors:
predictors =
['Income','age','PCIGS79','Incomesq','agesq','PCIGS79sq','PCIGS79sq']
xsqc = logit1[predictors]

   # Define dependent variable:
y = logit1['Smoker']

# Estimate the model:
logit2Reg = LogisticRegression()
lr=logit2Reg.fit(xsqc, y)

print(lr.coef_)

# Let's cross validate the results of the squared and linear terms model
above using k-10 validation
    # Define predictors
predictors = ['Income', 'age', 'PCIGS79', 'Incomesq', 'agesq', 'PCIGS79sq',
'IncomeagePCIGS79']
x2 = logit1[predictors]
    # Define target variable
y = logit1['Smoker']
    # Estimate the model
logit2Reg = LogisticRegression()
lr=logit2Reg.fit(xsqc, y)
    # Perform 10- fold cross validation
scoreslrsq = cross_val_score(lr, x2, y, cv=10)
print(scoreslrsq)
scoreslrsq.mean()

Mean score is: 0.6037394957983192
```

The confusion matrix provides a detailed breakdown of the model's performance in classifying smokers and non-smokers. True negative values of 627 represent the number of observations(individuals) that are non-smokers, as predicted by our model. On the other hand, the number of True positives (smokers) is only 107 observations. The model, however, incorrectly predicted 114 individuals who are non-smokers as smokers, representing them as false positives. Finally, 348 smokers were predicted not to be smokers, denoted as false negatives.

Model prediction

```
y_pred = cross_val_predict(lr, xsqc, y, cv=10)
conf_mat = confusion_matrix(y, y_pred)
    # create confusion matrix
cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix =
conf_mat, display_labels = [False, True])
fig, ax = plt.subplots(figsize=(6, 6))
cm_display.plot(ax=ax)
plt.title("Confusion matrix for smokers & non-smokers")
    # show the matrix
plt.show()
```



The high number of true negatives (627) suggests that the model is effective at identifying non-smokers. However, the significant number of false negatives (348) indicates that the model struggles to identify smokers accurately. This could imply that the model may require further refinement or the inclusion of additional features to enhance its predictive capability for smokers. The false positives (114) suggest that there are some non-smokers incorrectly classified as smokers, which could lead to unnecessary interventions or misinterpretations in a practical setting. Overall, while the model performs well in identifying non-smokers, it may require adjustments to improve its accuracy in identifying smokers.

Linear Discriminant Analysis

```
# define the variables:
predictor2 = ['Income', 'age', 'PCIGS79']
x2 = logit1[predictor2]
y2 = logit1['Smoker']
# Estimate the model:
ld2 = LinearDiscriminantAnalysis()

scoresld2 = cross_val_score(ld2, x2, y2,cv=10)
print(scoresld2)
scoresld2.mean()

Mean score is: 0.6003221288515406

logit1['Incomesq'] = logit1['Income']**2
logit1['agesq'] = logit1['age']**2
logit1['PCIGS79sq'] = logit1['PCIGS79']**2

# Interaction-terms:
logit1['IncomeagePCIGS79'] = logit1['Income']*logit1['age']*logit1['PCIGS79']

predictors3= ['Income', 'age', 'PCIGS79', 'Incomesq', 'agesq', 'PCIGS79sq',
'IncomeagePCIGS79']
x3 = logit1[predictors3]
y3 = logit1['Smoker']

ld3 = LinearDiscriminantAnalysis()
scoresld3 = cross_val_score(ld3, x3, y3,cv=10)
print(scoresld3)
scoresld3.mean()

Mean score is: 0.6079131652661063
```

Thus, mean cross-validation scores for both the logistic regression and linear discriminant analysis (LDA) models are similar, indicating that the models have a comparable ability to predict smoking status. This suggests that both modeling approaches are effective in capturing the underlying relationship between the predictors (Income, age, PCIGS79) and their associated target variable(smokers). Interestingly, the second models, thus, squared and interaction terms models in both the logistic regression and LDA, show a slight improvement in the mean cross-validation scores for both the logistic regression and LDA approaches. This suggests that incorporating non-linear relationships and interactions between the predictors may enhance the models' predictive performance for whether a person is a smoker or not.

The similar performance between the logistic regression and LDA models is noteworthy, as they make different assumptions about the underlying data distribution. Pohar et al. (2004) explained that logistic regression assumes a linear relationship between the predictors and the log-odds of the dependent variable, while LDA assumes a multivariate normal distribution of the predictors and equal covariance matrices across the classes (smokers and non-smokers). This suggests that the predictors have a strong influence on the dependent variable, and the models are able to effectively capture the patterns in the data, regardless of the specific assumptions made. The slight improvement observed in the second models, which incorporate non-linear terms, further highlights the potential presence of non-linear relationships or interactions between the predictors and the smoking status. This finding suggests that the inclusion of such non-linear terms may be beneficial for both the logistic regression and LDA models, potentially leading to enhanced predictive performance for smoking status.

Overall, the similar performance of the logistic regression and LDA models, along with the potential benefits of incorporating non-linear terms, provide valuable insights into the nature of the relationship between the predictors and the smoking status as the dependent variable. These findings can guide the selection and refinement of the most appropriate modeling approach for predicting smoking status in practical applications.

Part 2: Model selection

This dataset consists of yearly data collected by Swedish energy market inspectorate (SEMI), which represents Swedish electricity distribution system operators (DSOs), averaged over 2016-2019. This case is relevant to study since SEMI collects a substantial amount of technical, accounting and market data from the DSOs every year. Based on these data, SEMI determines the relevant cost drivers using a mixture of regression-based techniques and their technical knowledge of the energy distribution market. In 2019, the Swedish electricity distribution market consisted of 154 DSOs. The variable definitions of independent variables is available in Table 4.1 in appendix and the dataset is Modelselection2.csv. The dependent variable is Totex (total cost).

1. Find the optimal model using forward selection
2. Find the optimal model using Lasso.
3. Find the optimal model using an elastic net.
4. Comment on the result.

Forward selection technique

In the process of developing a robust predictive model for the target variable Totex, a linear regression model was constructed using the features selected through the forward feature selection technique. To assess capability of this model, a 10-fold cross-validation procedure was employed, with the mean squared error (MSE) serving as the evaluation metric.

Forward selection

```
 # Lets define the independent variables
x = sdf[sdf.columns.difference(['Totex','REid'])]
# Target variable
y = sdf['Totex']

# Create a linear regression model
lr1 = LinearRegression()

# Create a forward feature selector
ffs = SequentialFeatureSelector(lr1, n_features_to_select=None,
direction='forward')

# Fit the forward feature selector
```

```
ffs.fit(x, y)

# Get the selected features
selected_features = x.columns[ffs.get_support()]
print('Selected features:', selected_features)

# Fit the linear regression model using the selected features
final_model = LinearRegression()
final_model.fit(x[selected_features], y)

# Print the model coefficients
print('Model coefficients:', final_model.coef_)

x = sdf[['LVEnergy', 'HVenergy', 'LVSubs', 'HVSubs', 'Networks', 'ULVOL',
       'UHVOL', 'Temp', 'Industri', 'Agriculture', 'Growth']]
# Cross validate the forward selection model
scoreforwardsel = cross_val_score(final_model, x, y, cv=10,
scoring='neg_mean_squared_error')
print('MSEs of independent test sets:', abs(scoreforwardsel))

# Lets determine the average performance of the model
print('\n',abs(scoreforwardsel.mean()))

MSEs of independent test sets: [0.01464824 0.00473124 0.00555026 0.00642405
0.0059037  0.00212806
 0.00578993 0.02956129 0.06015928 0.01055511]

Mean score is: 0.014545116322859541
```

The cross-validation results provide valuable insights into the model's performance. The MSE scores obtained for each of the 10 folds represent the model's predictive accuracy on independent test sets, offering a reliable estimate of its ability to generalize to unseen data. By examining the MSE scores above, we can observe that the model is moderately consistent in making accurate predictions across the different subsets of the data.

This metric serves as a concise summary of the model's performance, clearly indicating its suitability for the given problem. A lower average NMSE score suggests that the linear regression model, in conjunction with the selected features, is capable of accurately predicting the target variable Total cost(Totex). Thus, comparing the results of the forward selection to Lasso and Elastic net, the forward selection technique is a poor predictor of total cost compared to Lasso. This could be so because Lasso regression and Elastic net are regularized linear regression techniques that performs feature selection by shrinking some coefficients to zero, which can lead to improved generalization performance whereas forward selection does not penalize variables, hence the underperformance of the forward selection model.

Find the optimal model using Lasso

We started by standardizing our dataset to normalize the features, ensuring they all have a mean of zero and a standard deviation of one. This step helps make different features comparable to Machine learning.

Then, we included all variables from the dataset except Totex and REid as predictors, as we can't use Totex as a predictor for Totex. We also excluded REid, as that variable organizes all other variables and thus has 0 connection with the different variables.

Then, we created a model to find the optimal Alpha using the following code.

```
ModelL = Lasso(alpha=1)
ModelL.fit(X1,Y1)

print('Model coefficients', ModelL.coef_)

ModelLcv = LassoCV(cv=10, random_state=0, max_iter=1000)

ModelLcv.fit(X1,Y1)
```

This model then produced an Alpha value of: 0.003983093748065897
Thus, we replace our precious alpha value of 1 with our new value of 0.003983093748065897.
And run a model looking for the Mean Squared Error of the model. Using the following code:

```
ModelL1 = Lasso(alpha = 0.003983093748065897)
ModelL1.fit(X1,Y1)

ScoreL1 = cross_val_score(ModelL1, X1, Y1, cv=10,
scoring='neg_mean_squared_error')
```

Producing a mean score of Mean Squared Error of : 0.013750812357836614

Find the optimal model using elastic net

Again, we start by Standardizing our variables and cosigning our Predictors (except Totex and REid). Then, we created a model to find the Mean Squared error using an alpha value of 1.

```
EL = ElasticNet(alpha=1.0, l1_ratio=0.5)
ModelEL = EL.fit(X2,Y2)

ScoreEL = cross_val_score(ModelEL,X2,Y2, cv=10,
scoring='neg_mean_squared_error')
```

After this, we created a loop that runs the same model with different l1_ratio values specified in an array. For each l1_ratio value, an ElasticNetCV model is created and fitted to the data, adjusting the regularization balance between L1 (Lasso) and L2 (Ridge). Each model's mean squared error (MSE) is then calculated using 10-fold cross-validation and printed.

```
for r in ratio:

    l1=r
    elr = ElasticNetCV(cv=10, l1_ratio=r,random_state=0 , max_iter=10000)
    modelelr = elr.fit(X2, Y2)

     scoreselr = cross_val_score(modelelr, X2, Y2,
cv=10,scoring='neg_mean_squared_error')

    MSEs= scoreselr.mean()
    print('\n',MSEs)
```

This approach helps evaluate the model's performance under various regularization settings, allowing us to determine the l1_ratio value that yields the lowest cross-validated MSE, indicating the best regularization balance.

This then gave us an average of about 0.015. Which we used as our new l1_ratio, when fitting the final Elastic Net model.

```
elr = ElasticNetCV(cv=10, l1_ratio=0.015, random_state=0)
```

```
modelelr = elr.fit(X2, Y2)

scoreselr = cross_val_score(modelelr, X2, Y2,
cv=10,scoring='neg_mean_squared_error')
```

Producing a Mean Squared Error of 0.022251021333842623

Comparing Results

The Mean Squared Error (MSE) produced by the Lasso model was 0.013750812357836614, while the Elastic Net model produced an MSE of 0.022251021333842623.

The difference between these scores indicates that the Lasso model performed better, as it has a lower MSE. A lower MSE suggests that the Lasso model had a better fit to the data, making more accurate predictions. This could be because Lasso regularization (L1 penalty) encourages sparsity in the model by shrinking some coefficients to zero, effectively selecting only the most important features.

In contrast, the Elastic Net model, which combines both L1 (Lasso) and L2 (Ridge) penalties, might not have been as effective in this particular case. The MSE difference suggests that the inclusion of the L2 penalty might not have contributed to a better fit, potentially due to the dataset characteristics or the choice of hyperparameters (like l1_ratio).

In summary, based on the MSE scores, the Lasso model was more effective than the Elastic Net model in minimizing prediction error for this specific dataset.

Post Lasso

Use the dataset Modelselection2.csv where the dependent variable is Totex (total cost). We wish to investigate the effect of temp on Totex. Transform the data into log-form. Perform Post-Lasso to investigate the effect. Comment on the result (is it significant effect etc.). Please hand in the final solution (relevant output) with commentary before 27th of September. Upload the final version on Canvas

We started by transforming the data into log form. Then, we picked out our predictors, all variables except Totex, REid, and Temp. Then, we ran a Lasso cross-validation for Totex and Templog separately. This gave us the following outputs:

```
  Post lasso Coef Totex
[-0.01573566  0.02684533  0.689682    0.26093278  0.          -0.
 -0.         -0.          0.00464446 -0.         -0.          0.04469645
  0.          0.          0.          0.         -0.00801748 -0.0498252
  0.        ]


   Post lasso Coef for Temp
[-0.08642217 -0.10373644  1.17766214  0.54668404 -1.00519012  0.04761059
  0.01623857 -0.00134168 -0.01169809  0.          0.00730919  0.
 -0.00915585 -0.09369243 -0.02461763  0.00604264 -0.00280296  0.1694815
 -0.063951  ]
```

Then, we eliminated all variables that showed a coefficient of 0 on both totes and temp, in this case only ULVOL.

After this, we performed an OLS regression on Totex using Temp as our predictor; this gives us a baseline model showcasing just how well a simple model with only one predictor would work. This will work as our control variable.

Then, we perform an OSL regression on Totex using all variables (except Totex, REid, and ULVOL). The goal here is to assess how these selected variables, which Lasso determined to be important, perform in an OLS model.

```
Modeling only Temp as predictor for Totex (Control model)
R-squared :             0.684
Adj. R-squared:         0.681
F-Statistic :                 308.9
AIC :                         247.6
BIC :                         253.6
```

```
Modeling all variables as predictors for Totex
R-squared :             0.984
Adj. R-squared:         0.982
```

```
F-Statistic :                    406.8
AIC :                                 -149.9
BIC :                                 -90.36
```

When the two models were compared, the R-squared value significantly increased from 0.684 to 0.984, indicating that Modeling all variables as predictors for Totex explains 98.4% of the variation in Totex, compared to only 68.4% in the Control model.

The F-statistic compares the two models and shows that Modeling all variables as predictors has a stronger overall fit than the Control model.

In the Control model, the F-statistic is 308.9, with a p-value of 1.49e-37, indicating that the model is highly statistically significant. This means that Temp alone strongly predicts Totex in this simplified model.

However, when all variables are modeled as predictors, the F-statistic increases to 406.8, with an even smaller p-value of 1.02e-102. The higher F-statistic indicates a better fit and more substantial explanatory power. This, combined with the extremely low p-value, confirms that the selected predictors after Lasso regularization are jointly highly significant in explaining Totex.

When the two models are compared, the AIC and BIC values show that modeling all variables as predictors shows a much better fit than the Control model.

In the Control model, the AIC is 247.6, and the BIC is 253.6. These values suggest a relatively poorer fit and indicate that the model's simplicity (only using Temp as a predictor) comes at the cost of not capturing enough complexity in the data.

Modeling all variables as predictors significantly lowers the AIC value to -149.9 and the BIC to -90.36. These negative values suggest a superior fit. The large reduction in both AIC and BIC implies that the Post-Lasso model strikes a better balance between model complexity and goodness-of-fit.

References

Pohar, M., Blas, M., & Turk, S. (2004). Comparison of logistic regression and linear discriminant analysis. *Advances in Methodology and Statistics*, *1*(1), 143–161. https://doi.org/10.51936/ayrt6204