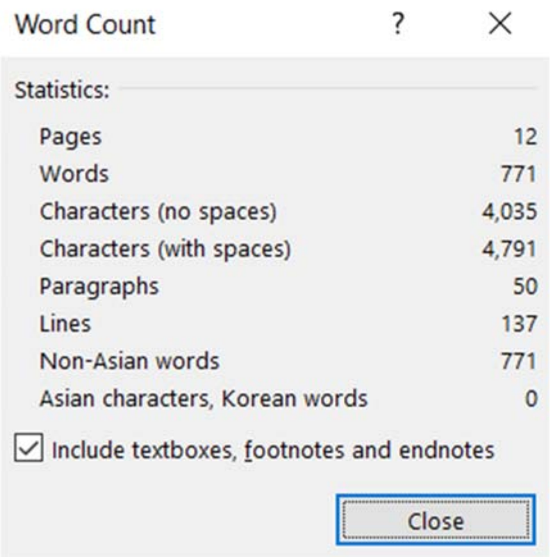
KSD DM Mandatory Assignment 1



Word Count

Statistics:

Pages	12
Words	771
Characters (no spaces)	4,035
Characters (with spaces)	4,791
Paragraphs	50
Lines	137
Non-Asian words	771
Asian characters, Korean words	0

☑ Include textboxes, footnotes and endnotes

IT UNIVERSITY OF COPENHAGEN

OCTOBER 25

IT University of Copenhagen
Authored by: Asger Schliemann-Haug
IT-email: ascs@itu.dk

Experimenting on micro-scale data mining

Introduction with goals for the experiment

The reports seek to investigate different data mining aspects on a dataset generated by an ITU questionnaire, answered by 50 students, with the schema as follows (transposed for the sake of the overview):

Timestamp
Your mean shoe size (In European Continental system)
Which programme are you studying?
Your height (in International inches)
Why are you taking this course?

The source code along with the dataset, can be found in the non-private github repository: https://github.com/asgerhaug/Data-Mining-Assignment-1.git

The sample has 50 rows of entries. For which the author selected the following questions to answer as best as possible:

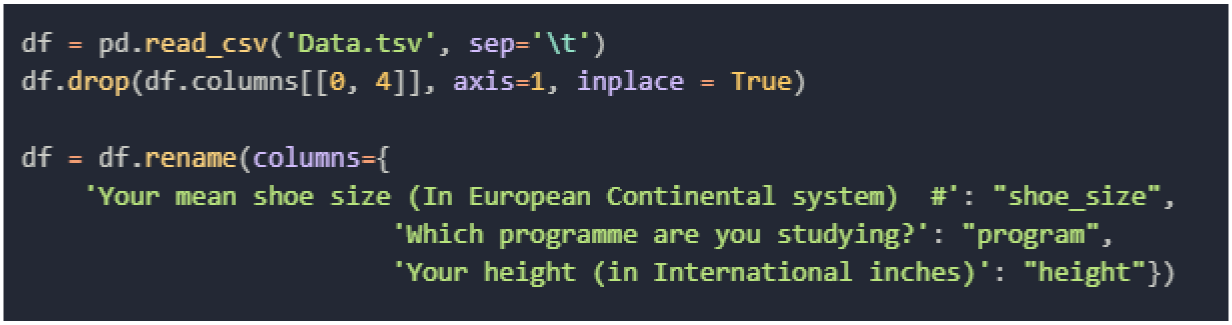
Question A) *"Is there a correlation between the shoe size, program chosen and one's height? If yes, how well can a Gaussian Naive Bayes supervised learning model predict one's height based on the shoe size and the chosen study programme?*

Question B) *"Is it possible by generating an attribute of male and female (gender), to view them as clusters based on the input on gender, height, and shoe size? If yes, how does the relation between male and female stand against the same relation taken from the course participants (external data, not supplied in the schema –to make it more interesting).*

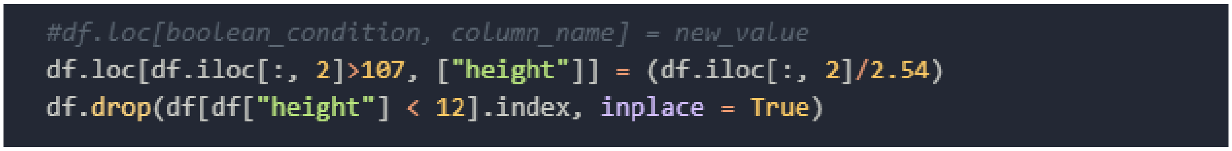
Cleaning the data

Initially, the data was cleaned, which involves the following steps:

1. Dropped the columns which were considered non-relevant
2. Renaming the columns to shorter Strings to ease the future code scripting
3. Converting what could be assumed to be wrongly inserted values due to simple type errors, i.e. people inserting their height in cm instead of inches.
4. Dropping rows where we cannot be sure why the entry is faulty, i.e. people inserting their shoe size as 9, where the smallest kid-size seems to be 18.[1]



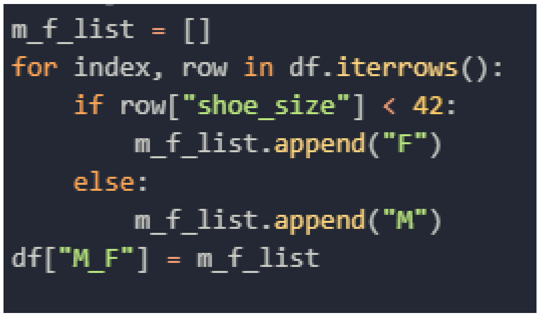
```
df = pd.read_csv('Data.tsv', sep='\t')
df.drop(df.columns[[0, 4]], axis=1, inplace = True)

df = df.rename(columns={
    'Your mean shoe size (In European Continental system)  #': "shoe_size",
                       'Which programme are you studying?': "program",
                       'Your height (in International inches)': "height"})
```



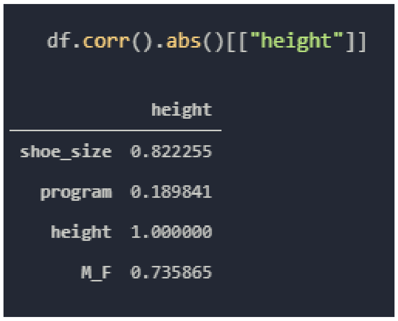
```
#df.loc[boolean_condition, column_name] = new_value
df.loc[df.iloc[:, 2]>107, ["height"]] = (df.iloc[:, 2]/2.54)
df.drop(df[df["height"] < 12].index, inplace = True)
```

Creating an attribute

The assumption was made that if one's shoe size is greater than 42, then one is a male. This is of course not correct in real life. It was however implemented for this experiment to introduce additional categorial data.



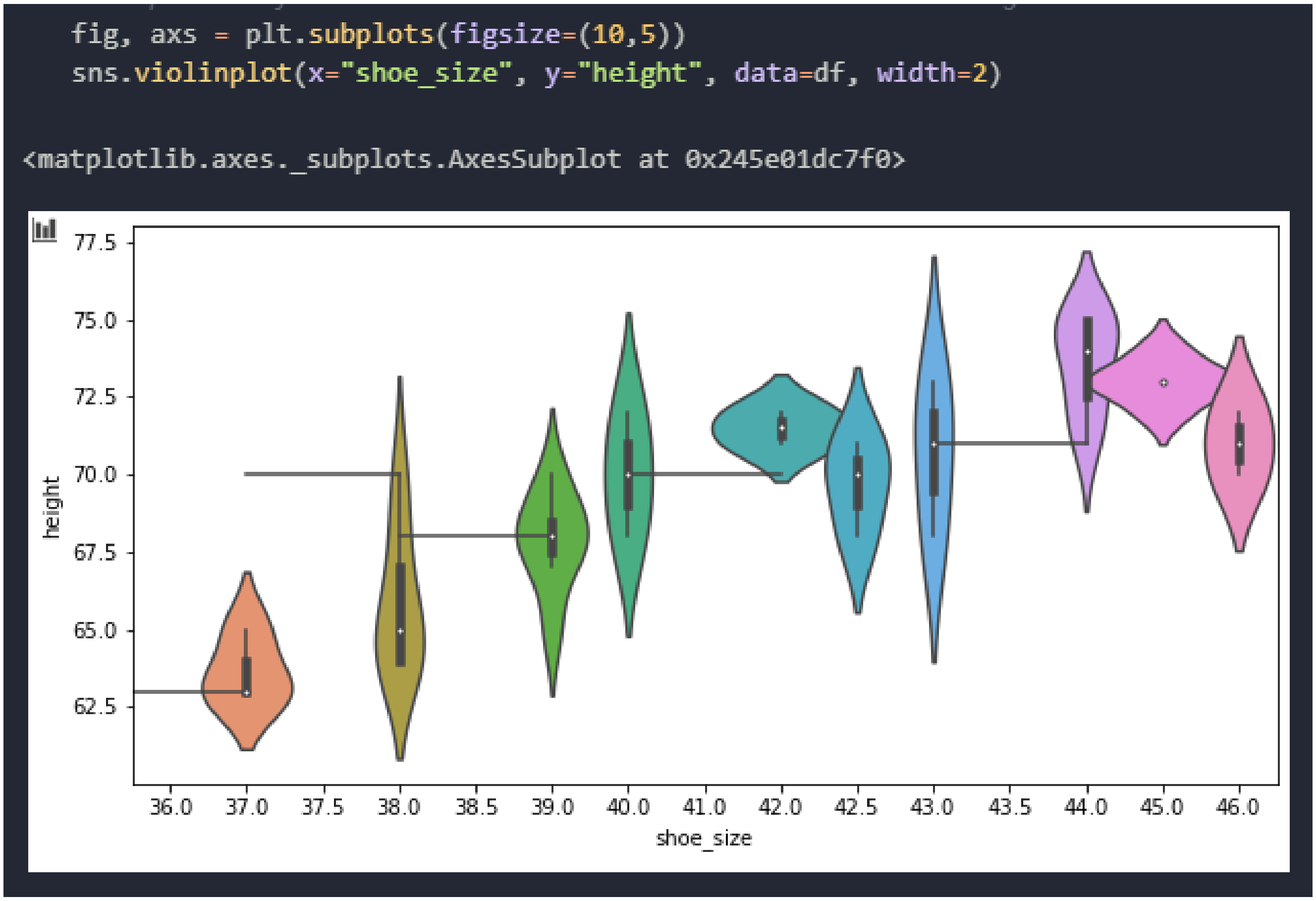
```
m_f_list = []
for index, row in df.iterrows():
    if row["shoe_size"] < 42:
        m_f_list.append("F")
    else:
        m_f_list.append("M")
df["M_F"] = m_f_list
```



```
df.corr().abs()[["height"]]

              height
shoe_size   0.822255
  program   0.189841
   height   1.000000
      M_F   0.735865
```

Taking a first glance at the correlations and the data plot of the biggest correlator.

From the correlation, it appears that shoe size the correlation factor of 0.822. This is somehow evident of the graphed subplot as well.

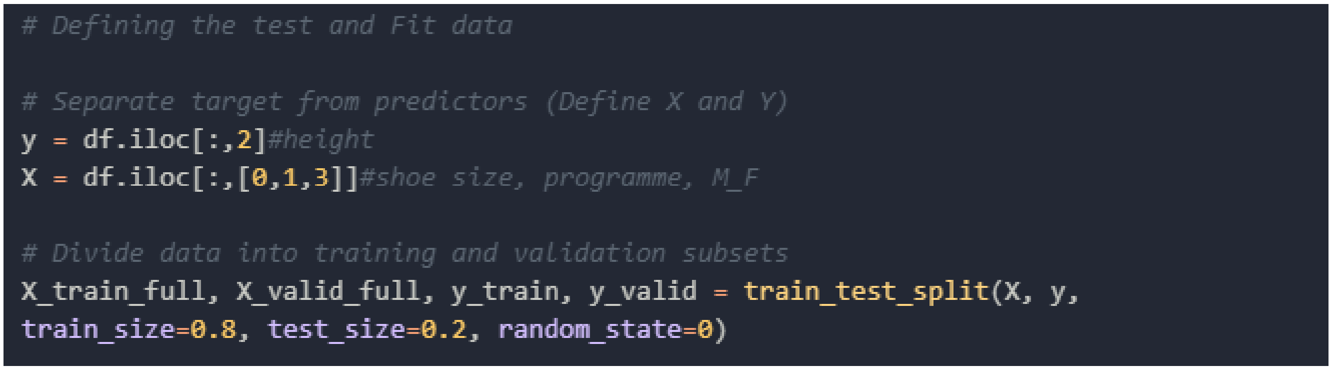


```
    fig, axs = plt.subplots(figsize=(10,5))
    sns.violinplot(x="shoe_size", y="height", data=df, width=2)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x245e01dc7f0>
```

Defining the test and fit data:

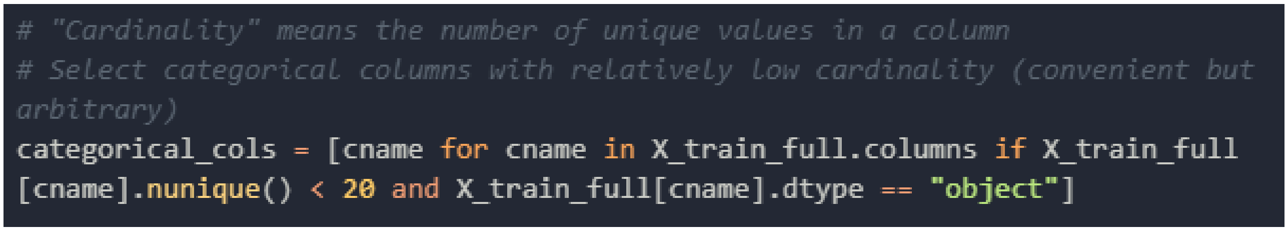
Using sklearn.model_selection module, the training size of the data was set to 80%



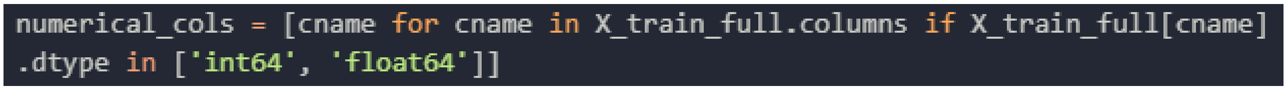
```
# Defining the test and Fit data

# Separate target from predictors (Define X and Y)
y = df.iloc[:,2]#height
X = df.iloc[:,[0,1,3]]#shoe size, programme, M_F

# Divide data into training and validation subsets
X_train_full, X_valid_full, y_train, y_valid = train_test_split(X, y,
train_size=0.8, test_size=0.2, random_state=0)
```

Preparing data for preprocessing pipeline:

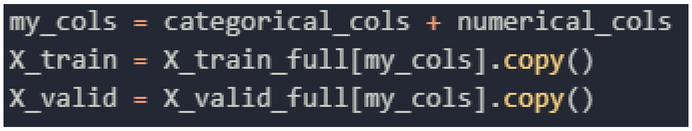
Preprocessing methods were required to be applied in the assignment. Therefore, the data was prepared for later pipelining by defining the columns as numeric or categorial.



```
# "Cardinality" means the number of unique values in a column
# Select categorical columns with relatively low cardinality (convenient but
arbitrary)
categorical_cols = [cname for cname in X_train_full.columns if X_train_full
[cname].nunique() < 20 and X_train_full[cname].dtype == "object"]
```

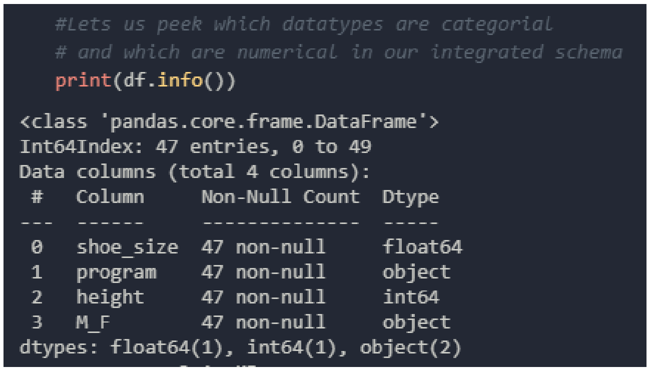


```
numerical_cols = [cname for cname in X_train_full.columns if X_train_full[cname]
.dtype in ['int64', 'float64']]
```

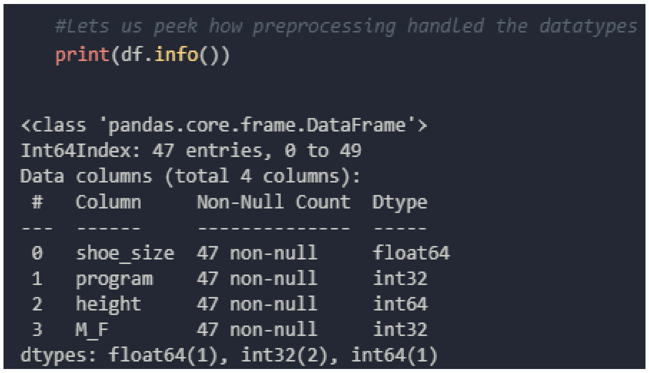


```
my_cols = categorical_cols + numerical_cols
X_train = X_train_full[my_cols].copy()
X_valid = X_valid_full[my_cols].copy()
```

The two print statements show the data types before preprocessing and after preprocessing.

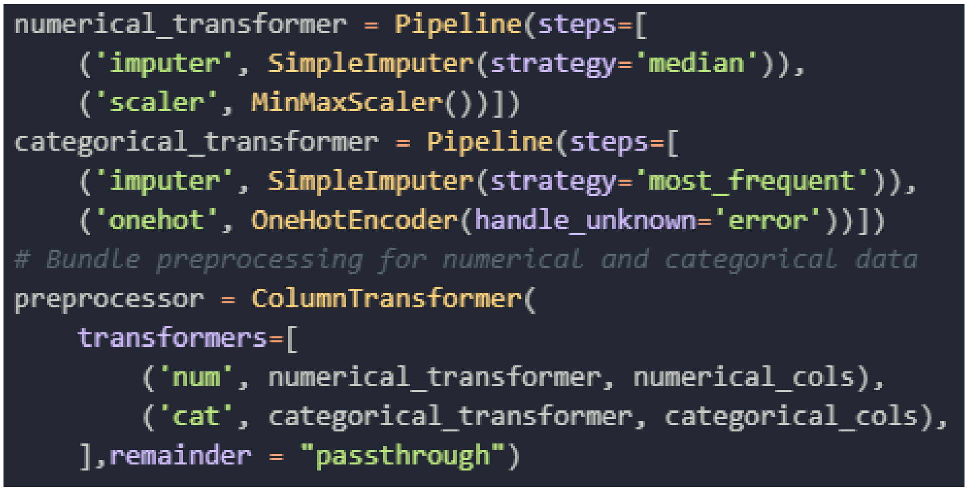


```
    #Lets us peek which datatypes are categorial
    # and which are numerical in our integrated schema
    print(df.info())

<class 'pandas.core.frame.DataFrame'>
Int64Index: 47 entries, 0 to 49
Data columns (total 4 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   shoe_size  47 non-null     float64
 1   program    47 non-null     object 
 2   height     47 non-null     int64  
 3   M_F        47 non-null     object 
dtypes: float64(1), int64(1), object(2)
```



```
    #Lets us peek how preprocessing handled the datatypes
    print(df.info())


<class 'pandas.core.frame.DataFrame'>
Int64Index: 47 entries, 0 to 49
Data columns (total 4 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   shoe_size  47 non-null     float64
 1   program    47 non-null     int32  
 2   height     47 non-null     int64  
 3   M_F        47 non-null     int32  
dtypes: float64(1), int32(2), int64(1)
```

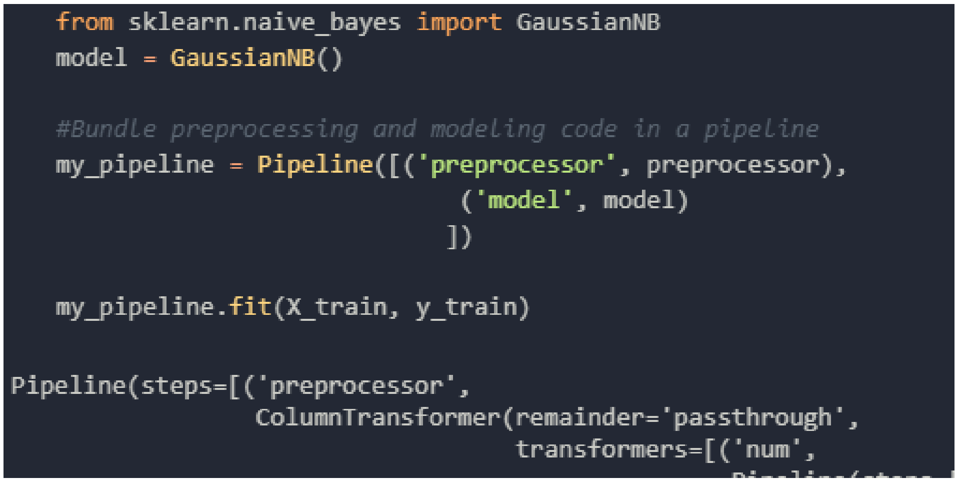
Implementing the ColumnTransformer for bundle preprocessing.



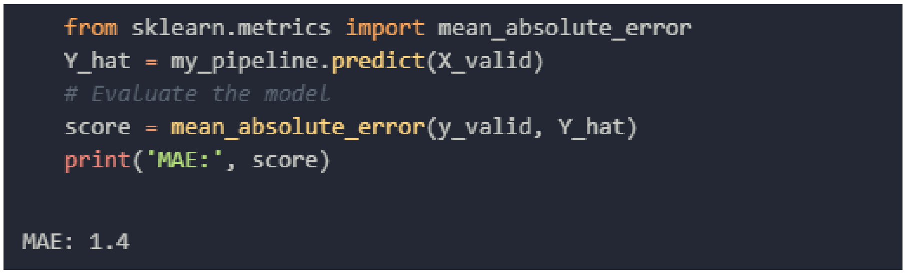
```
numerical_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='median')),
    ('scaler', MinMaxScaler())])
categorical_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='most_frequent')),
    ('onehot', OneHotEncoder(handle_unknown='error'))])
# Bundle preprocessing for numerical and categorical data
preprocessor = ColumnTransformer(
    transformers=[
        ('num', numerical_transformer, numerical_cols),
        ('cat', categorical_transformer, categorical_cols),
    ],remainder = "passthrough")
```

Implementing the supervised learning as Gaussian Naive Bayer's:

The model was decided as a Gaussian Naive Bayers. The Mean Absolute Error was 1.4

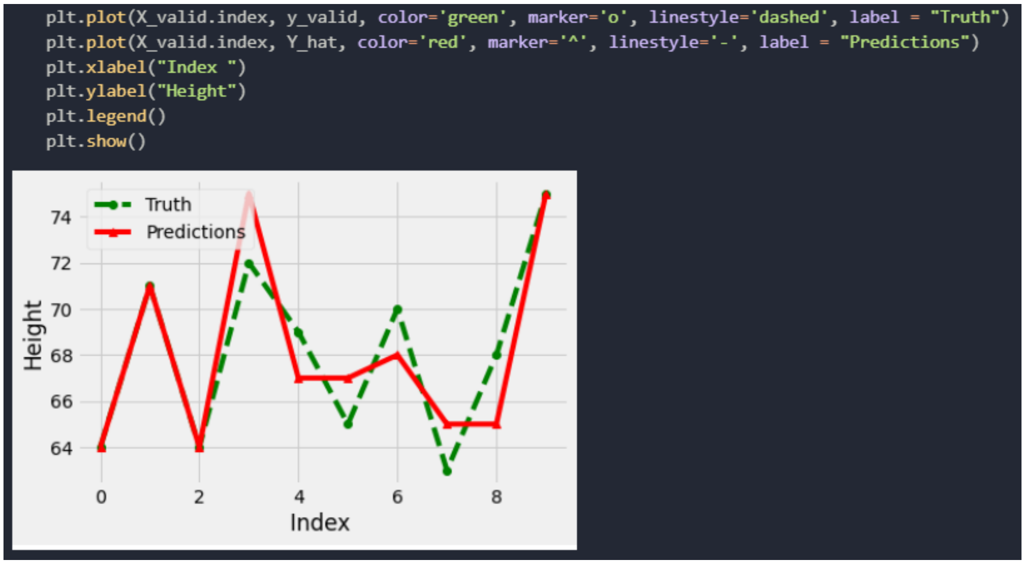


```
   from sklearn.naive_bayes import GaussianNB
   model = GaussianNB()

   #Bundle preprocessing and modeling code in a pipeline
   my_pipeline = Pipeline([('preprocessor', preprocessor),
                           ('model', model)
                          ])

   my_pipeline.fit(X_train, y_train)

Pipeline(steps=[('preprocessor',
                 ColumnTransformer(remainder='passthrough',
                                   transformers=[('num',
                                                  Pipeline(steps=
```



```
   from sklearn.metrics import mean_absolute_error
   Y_hat = my_pipeline.predict(X_valid)
   # Evaluate the model
   score = mean_absolute_error(y_valid, Y_hat)
   print('MAE:', score)

MAE: 1.4
```

The graphed "Predictions" vs "Truth" on the validation set (index). Note that "Truth" does not imply that the data is **true.** The author could not find a better word.

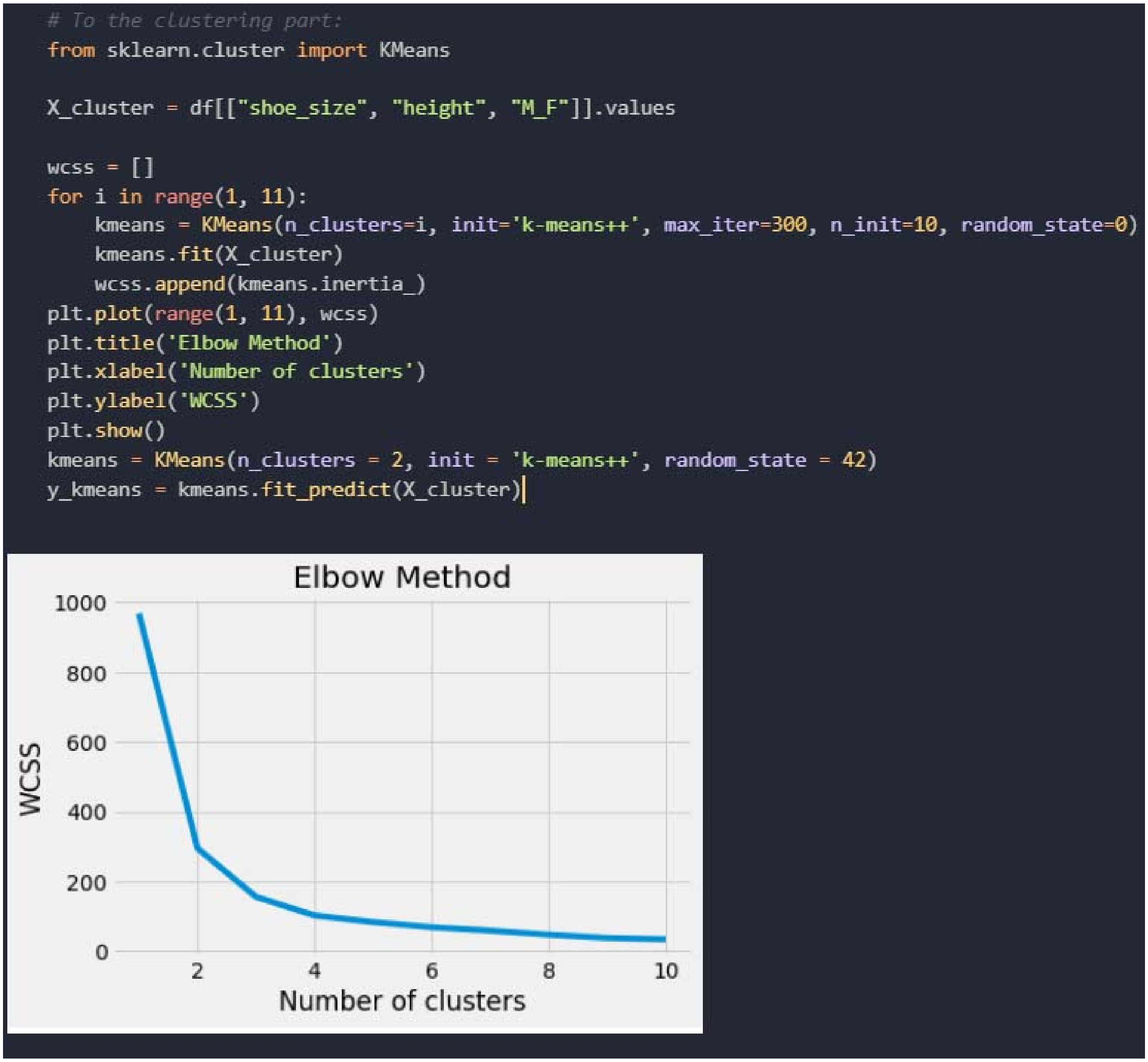


```
   plt.plot(X_valid.index, y_valid, color='green', marker='o', linestyle='dashed', label = "Truth")
   plt.plot(X_valid.index, Y_hat, color='red', marker='^', linestyle='-', label = "Predictions")
   plt.xlabel("Index ")
   plt.ylabel("Height")
   plt.legend()
   plt.show()
```

Conclusion to question A: The model can predict to some extent, but not well. It might be that the author has condensed the dataset to a more "regressional" nature, for which Naive Bayer's is not the best solution.

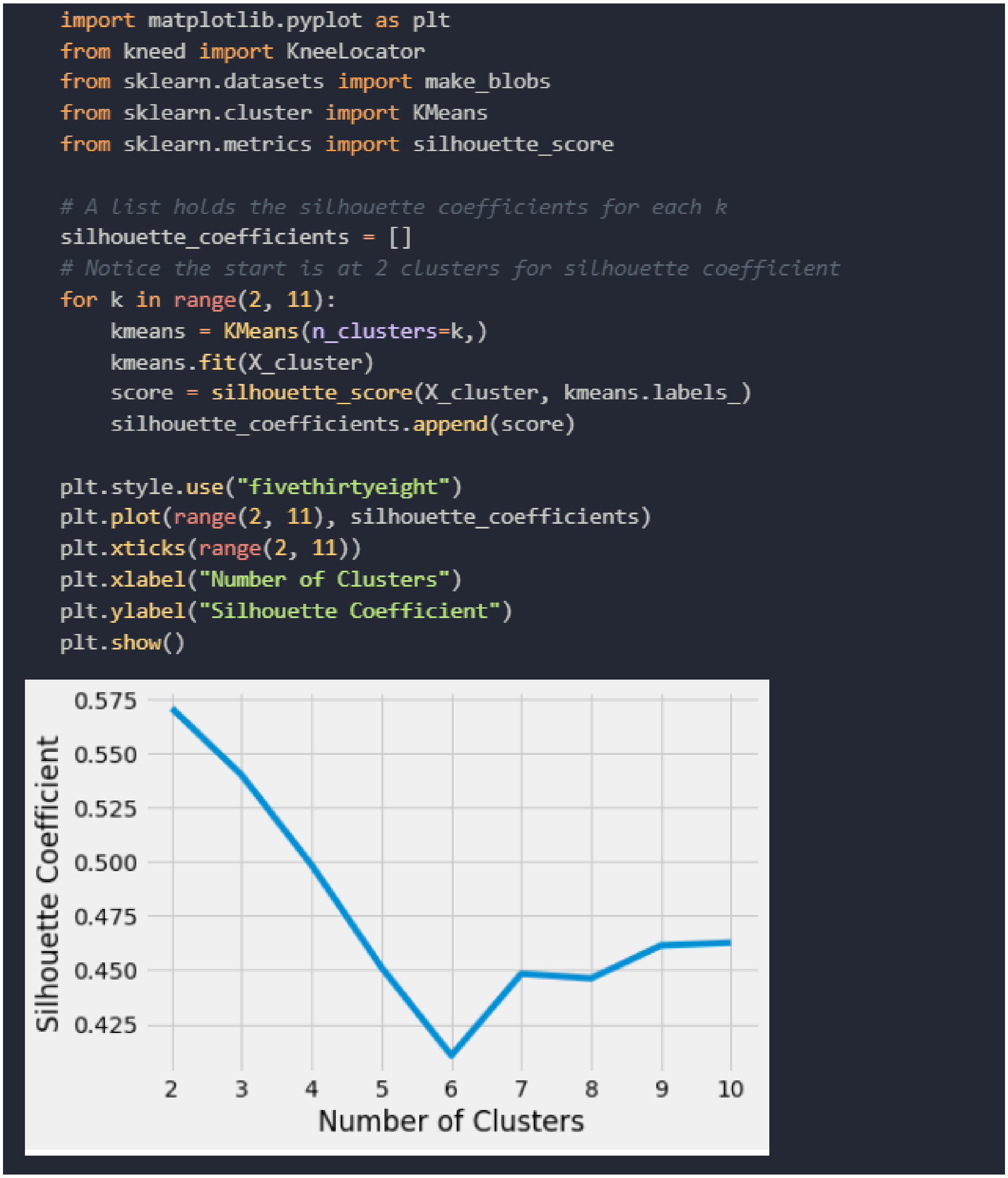
Clustering with K-means:

The Clustering was implemented with K-means, and the elbow method was used to find the ideal elbow-point of clusters, here being 2.



```
# To the clustering part:
from sklearn.cluster import KMeans

X_cluster = df[["shoe_size", "height", "M_F"]].values

wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, init='k-means++', max_iter=300, n_init=10, random_state=0)
    kmeans.fit(X_cluster)
    wcss.append(kmeans.inertia_)
plt.plot(range(1, 11), wcss)
plt.title('Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
kmeans = KMeans(n_clusters = 2, init = 'k-means++', random_state = 42)
y_kmeans = kmeans.fit_predict(X_cluster)
```

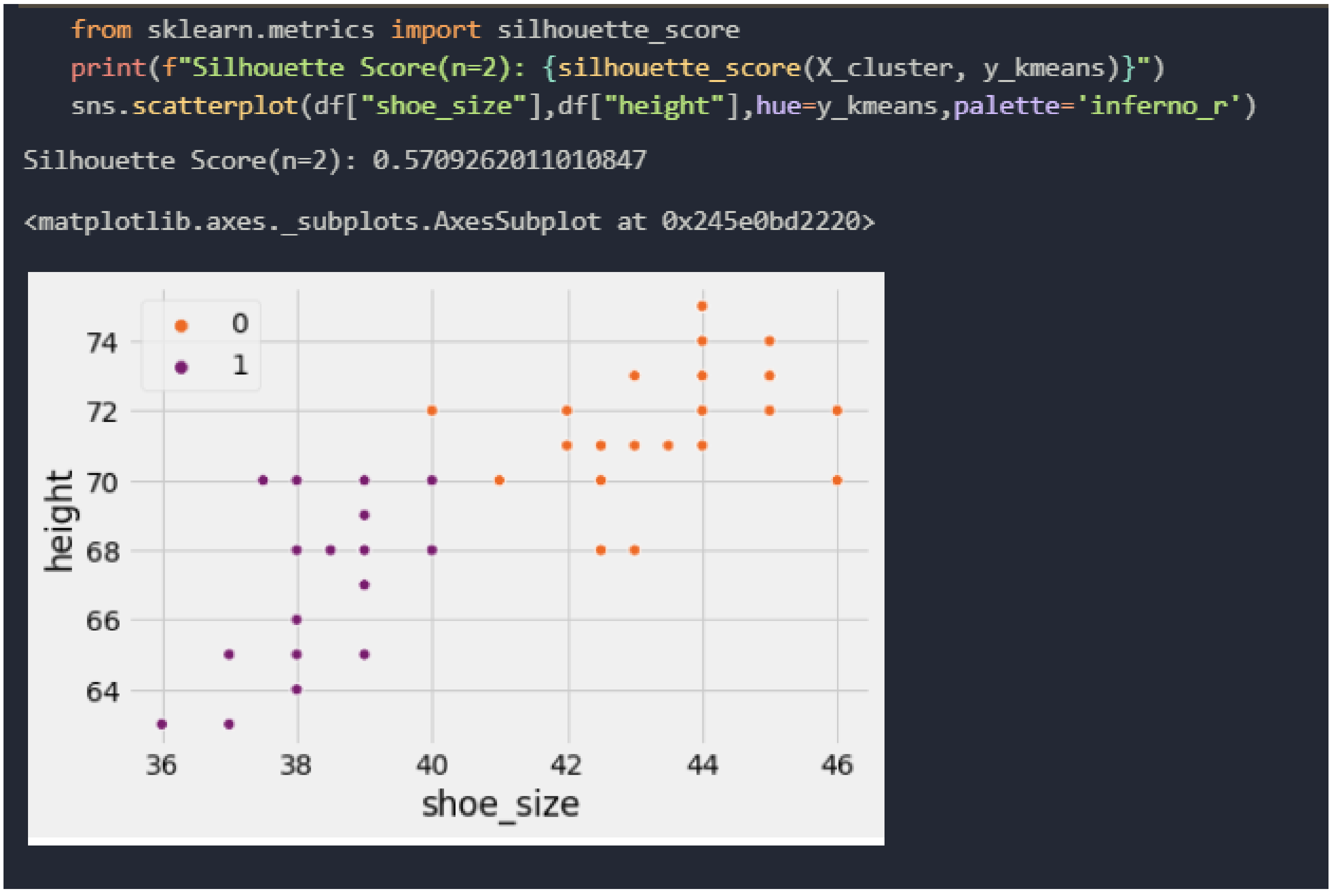
The silhouette coefficient also indicated its highest value at 2 clusters.



```
import matplotlib.pyplot as plt
from kneed import KneeLocator
from sklearn.datasets import make_blobs
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score

# A list holds the silhouette coefficients for each k
silhouette_coefficients = []
# Notice the start is at 2 clusters for silhouette coefficient
for k in range(2, 11):
    kmeans = KMeans(n_clusters=k,)
    kmeans.fit(X_cluster)
    score = silhouette_score(X_cluster, kmeans.labels_)
    silhouette_coefficients.append(score)

plt.style.use("fivethirtyeight")
plt.plot(range(2, 11), silhouette_coefficients)
plt.xticks(range(2, 11))
plt.xlabel("Number of Clusters")
plt.ylabel("Silhouette Coefficient")
plt.show()
```

The Cluster presented

The cluster shows two groups, which in question B, would imply Female and Male characteristics. Cluster 0 being male, and Cluster 1 being female, at a ration of 17 / 21.



```
    from sklearn.metrics import silhouette_score
    print(f"Silhouette Score(n=2): {silhouette_score(X_cluster, y_kmeans)}")
    sns.scatterplot(df["shoe_size"],df["height"],hue=y_kmeans,palette='inferno_r')

Silhouette Score(n=2): 0.5709262011010847

<matplotlib.axes._subplots.AxesSubplot at 0x245e0bd2220>
```

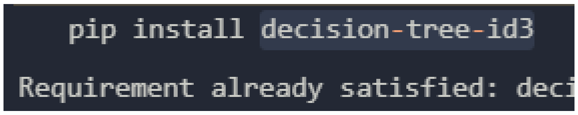
Using the course's webpage "participants", listing the 60 participants in a spreadsheet (attached as appendix A) and quick counting the male and females, excluding instructors and teaching assistants. The relationship is 28 / 29 (Female vs. Male). There could be many reasons why the relation is not the same as the clusters such as but no limited to:

1. 60 participants, but only 50 rows in the dataset
2. Some rows got dropped due to reasons described in the Data Cleaning Section,
3. The Female, Male attribute is a "fictive calculated" one.

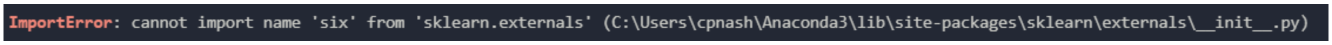
Conclusion to question B: The Cluster shows a pattern of 2 clusters, which is considered to be male and female.

Challenges and Frustrations

The ID3 fit model could not be implemented due to an annoying import error. Even though all the required packages and modules where installed.



```
    pip install decision-tree-id3

 Requirement already satisfied: deci
```



```
ImportError: cannot import name 'six' from 'sklearn.externals' (C:\Users\cpnash\Anaconda3\lib\site-packages\sklearn\externals\__init__.py)
```

Well… Character Limit Reached (4800).

[1] https://www.shooz4kidz.com/size-guide.html - viewed at 25-10-2020 16.09 GMT+1