IFT_6758_HW_2_(En)

November 8, 2020

IFT-6758 : Data Science

Fall - 2020

Homework - 2 - Matteo Esposito

Notebook due November 06, 2020 at 23.59 EST as **PDF** on Gradescope

[1]:
```
#@title Imports (Run this cell first) { run: "auto" }
plotting_library = "matplotlib"
import pandas as pd
import numpy as np

from sklearn.decomposition import PCA
from sklearn.linear_model import LinearRegression

# Not mandatory to use
from sklearn.utils import resample

import matplotlib.pyplot as plt

import seaborn as sns

# Uncomment this line below if using seaborn
sns.set()

%matplotlib inline

path = 'https://raw.githubusercontent.com/Jhelum-Ch/DataScience_IFT6758/gh-pages/
 ↪media/{}'
```

0.1 PCA

Q1 12 points = $(1.5 + 2 + 2 + 1.5 + 1.5 + 2 + 1.5)$

The cell below loads a subset of the California Housing dataset.

(a) Store only the `latitude`, `longitude` and `median_house_value` columns in a dataframe denoted by a variable `features`. Produce a scatter plot of the data points with `longitude` along x-axis, `latitude` along y-axis and the points colored by `median_house_value` i.e. higher the `median_house_value`, darker the data point in the plot.

(b) Perform a PCA on the subset of the dataframe you created in (a) with only the `latitude` and `longitude` columns. Produce a scatter plot of the transformed data points with the first principal component `PC 1` along x-axis and second principal component `PC 2` along y-axis and the points colored by `median_house_value` just like in (a).

(c) Provide a simple interpretation for what the first principal component `PC 1` could possibly represent in the plot in (b) by comparing it with that in (a). **Justify** your answer.

(d) Repeat what you did in (b) above by setting the `whiten` parameter as `True` in the `PCA()` constructor and producing the plot. What difference do you observe? What do you think `whiten` does specifically in this problem?

(e) Perform a PCA on the entire dataframe `features` with `whiten` set to `True` and produce a scatter plot of the transformed data points with the first principal component `PC 1` along x-axis and second principal component `PC 2` along y-axis and the points colored by `median_house_value` just like in (a).

(f) Observe how the color indicating `median_house_value` varies in the plot you produced in (e). Is the variation of `median_house_value` depicted in this plot simpler than what is indicated by all the above plots? Provide an **explanation** for why it is (or) it is not the case.

(g) The California Department of Housing and Community Development (HCD) releases additional information about the data samples you used here, by providing an `price_index` tag that can take values `high`, `middle` or `low` based `median_house_value`. If you were to eventually use the principal components you produced in (e), which one(s) among the `PC 1`, `PC 2`, ..,. would you use to classify the data samples into these three categories (`high`, `middle` and `low`)? Concretely **justify** your choice.

Bonus : (3 points) > (h) In the plot that you observe in (a), you will remark two major clusters that are the *darkest*. Let us identify the cluster with the higher value of `latitude` as the `SF cluster` and the one with the lower value of `latitude` as the `LA cluster`. Verify programmatically if this clustering is preserved or distorted in the plot in (e). What does this tell you about what is represented by the second principal component `PC 2` produced in (e)?

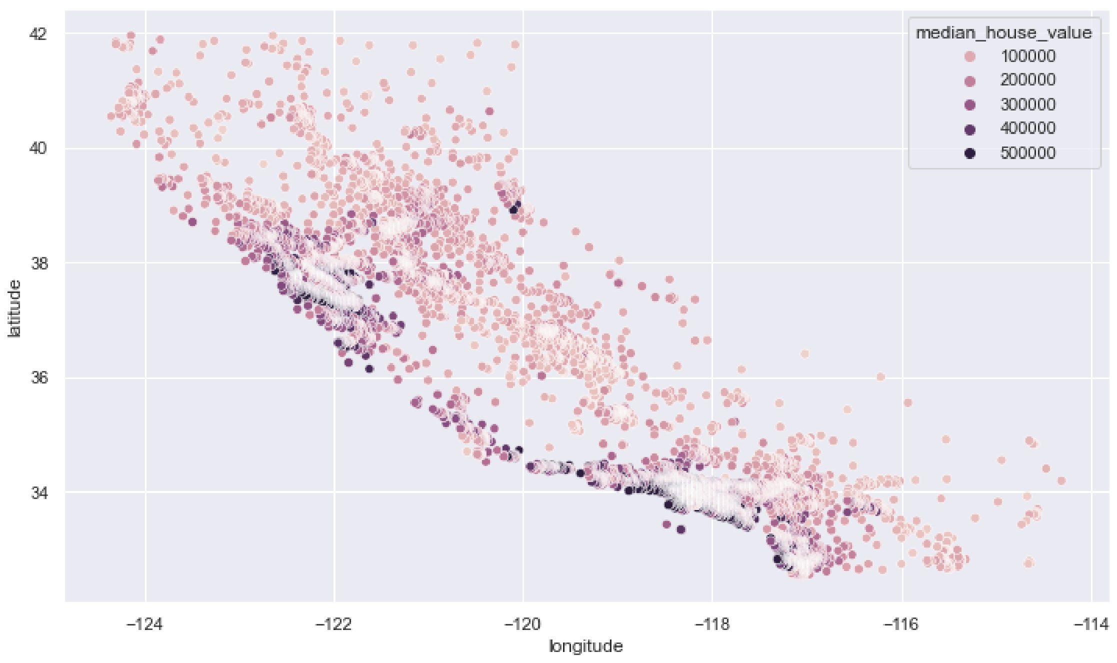
[2]:
```
housing = pd.read_csv(path.format('california_housing.csv'))
```

(a)

[133]:
```
features = housing[['latitude', 'longitude', 'median_house_value']]
features.head()
```

[133]:
```
   latitude  longitude  median_house_value
0     34.19    -114.31             66900.0
1     34.40    -114.47             80100.0
2     33.69    -114.56             85700.0
3     33.64    -114.57             73400.0
4     33.57    -114.57             65500.0
```

[134]:
```
plt.figure(figsize=(12,7))
sns.scatterplot(data=features, x='longitude', y='latitude',␣
 ↪hue='median_house_value');
```



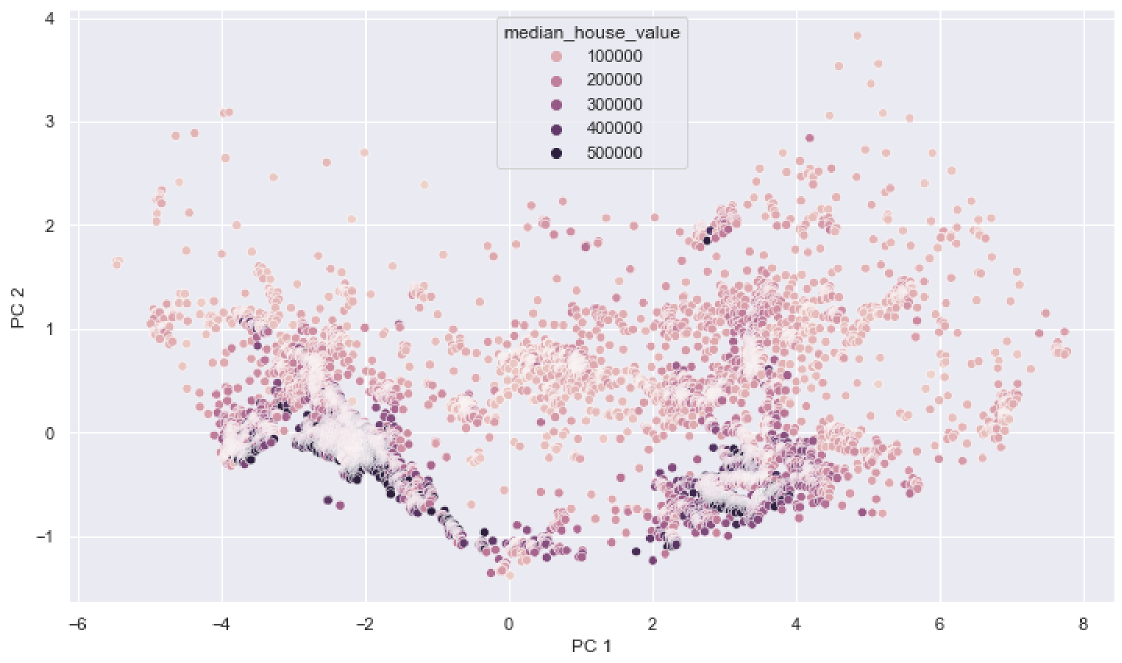
(b)

```
[138]: pca = PCA(n_components=2)
       pcs = pca.fit_transform(features[['latitude', 'longitude']])
       pca_df = pd.DataFrame(data=pcs, columns = ['PC 1', 'PC 2'])
       pca_df['median_house_value'] = features['median_house_value']
```

```
[139]: pca_df.head()
```

```
[139]:        PC 1      PC 2  median_house_value
       0 -4.632846  2.860289             66900.0
       1 -4.370153  2.886607             80100.0
       2 -4.827792  2.336365             85700.0
       3 -4.857522  2.294939             73400.0
       4 -4.908696  2.247176             65500.0
```

```
[140]: plt.figure(figsize=(12,7))
       sns.scatterplot(data=pca_df, x='PC 1', y='PC 2', hue="median_house_value");
```



(c)

PC1 likely represents the downward slope in the original dataset, from top left to bottom right. In other words, it represents the decrease in latitude and increase in longitude (this can be translated to the south-western-ness of a point).

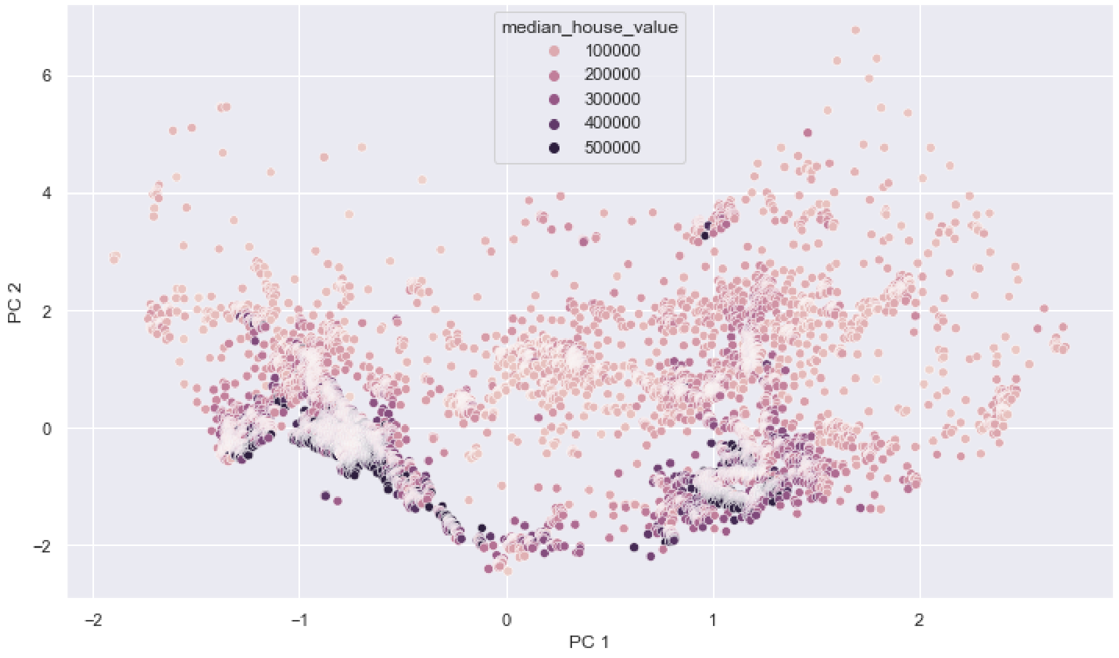
(d)

```
[8]: pca = PCA(n_components=2, whiten=True)
     pcs = pca.fit_transform(features[['latitude', 'longitude']])
     pca_df = pd.DataFrame(data=pcs, columns = ['PC 1', 'PC 2'])
     pca_df['median_house_value'] = features['median_house_value']
```

```
[9]: pca_df.head()
```

```
[9]:        PC 1      PC 2  median_house_value
     0 -1.611088  5.057651             66900.0
     1 -1.519736  5.104189             80100.0
     2 -1.678881  4.131233             85700.0
     3 -1.689220  4.057982             73400.0
     4 -1.707015  3.973527             65500.0
```

```
[10]: plt.figure(figsize=(12,7))
      sns.scatterplot(data=pca_df, x='PC 1', y='PC 2', hue="median_house_value");
```



Whitening in this example seems to contribute to variance reduction. The range of our PCs are smaller.

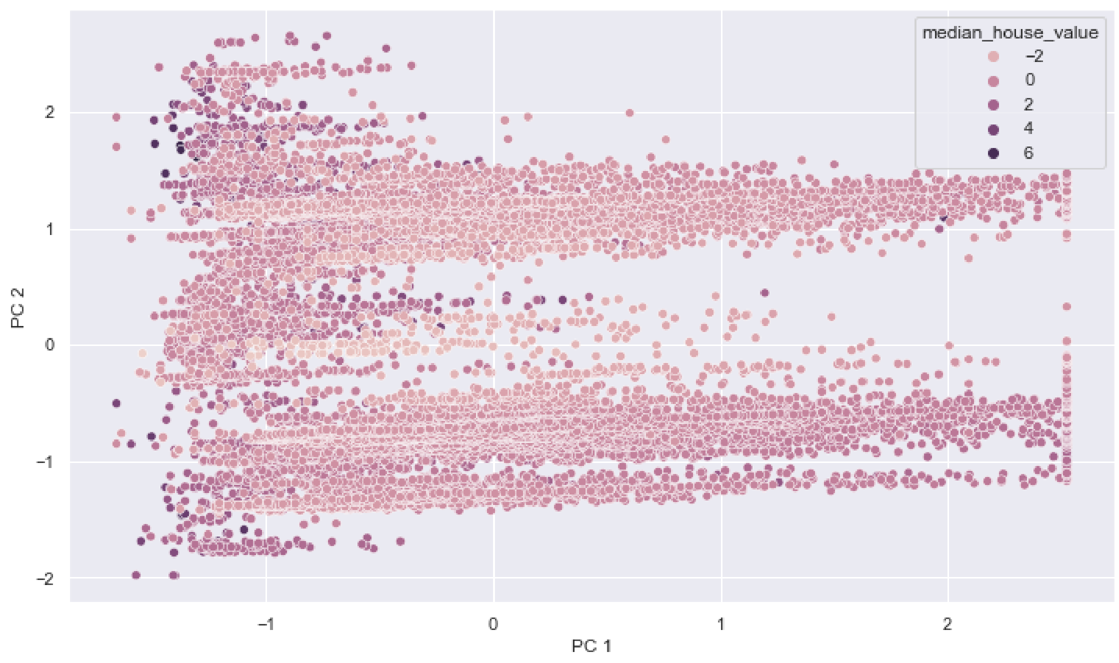
(e)

```
[11]: pca = PCA(n_components=3, whiten=True)
      pcs = pca.fit_transform(features)
      pca_df = pd.DataFrame(data=pcs, columns = ['PC 1', 'PC 2', 'median_house_value'])
```

```
[12]: pca_df.head()
```

```
[12]:        PC 1      PC 2  median_house_value
      0 -1.210522 -1.688243            5.068627
      1 -1.096713 -1.590321            5.189372
      2 -1.048430 -1.745882            4.094409
      3 -1.154480 -1.762187            3.950082
      4 -1.222593 -1.783791            3.814070
```

```
[13]: plt.figure(figsize=(12,7))
      sns.scatterplot(data=pca_df, x='PC 1', y='PC 2', hue="median_house_value");
```



(f) The variation in the median house value in the full dataset PCA case above is not simpler than the previous 2 PC case. It isn't simpler because there are multiple distinct ranges of PC1 and PC2 combinations that equate to the same median house value (e.g. median house value of 0 and 2 can be seen in PC1 ranges [-1, 2] and PC2 ranges [-1.5, -0.5] + [0.5, 1.5]) Whereas in the previous example, there is a gradual transition in median house values over the range of PCs.

(g) I would use PC2. This is because the median house values shown in the graph in the 2 PC example are easily distinguishable when referring to the y-component (PC2). We can get a rough but somewhat accurate idea of the price group from it alone. Compared to the PC1 axis which has one of each price groups (low, med, high) for each level in its entire range.

0.2 Clustering

Q2 10 points = $(1 + 2 + 3 + 1.5 + 1.5 + 2)$

A bird detection system that is equipped with multiple sensors is deployed in an observation station in an open space and it detects and collects some information about birds that visit the space. For each detected bird, it is able to collect the following information:

- `position` - a value of the form (x,y,z) which indicates the coordinates in a three-dimensional space defined by the LiDAR field of the system. `x` and `y` coordinates are in the range (-500,500) whereas `z` coordinate is in the range (0,150) where 0 indicates the ground level for `z`.
- `sound_level` - a value in decibels (0-120 dB) of the sound made by the detected bird with 0 indicating no audible sound and 120 indicating maximum sound that can be detected.
- `time_of_visit` - a value indicating number of milliseconds since the 00:00 hours of the day of collection.

The system sends out a table with the above three fields periodically to you. You are leading a team of data scientists to explore if the data collected by this system can be used to distinguish the different species of birds visiting the station located in the open space.

(a) You decide that using clustering for initially attempting this problem is a good option. How would you *logically* convince your team of this?

(b) Your team is trying to decide between using K-means and hierarchical clustering, both based on an Euclidean distance measure. Propose a list of preprocessing operations to be performed on the before you use any clustering algorithm on the raw dataset. **Justify** why you include each step.

(c) An enthusiastic intern in your team selects a subset of *raw* data samples from the dataset, selecting some nocturnal birds that visit all together exactly at midnight everyday and they cannot produce any sound. They all have been observed to consistently sit on the ground along a straight line. He generates the following dendrogram (left) using an agglomerative hierarchical clustering with an appropriate linkage that maximizes intercluster dissimilarity. On the right is a top-view visualization of the arrangement of the birds in the station, based on the dendrogram. The bird B1 has already been placed. Place the birds B2 - B9 on the line with an *appropriate* spacing between them. **Explain** your choice.

(d) You are informed by a group of expert ornitholigists that groups of birds that belong to different species, tend to visit the station in almost equal numbers per species. Based on this information, what type of linkage would you use in the hierarchical clustering algorithm? **Justify** your choice.

(e) The group of ornithologists have identified that exactly 10 species of birds visit the station. Does this information help you decide between choosing the K-means and hierarchical clustering algorithms? **Explain** why/why not.

(f) Briefly outline any **two** ways in which you can verify if the clustering you have performed has captured the natural grouping that exists among the actual data samples.

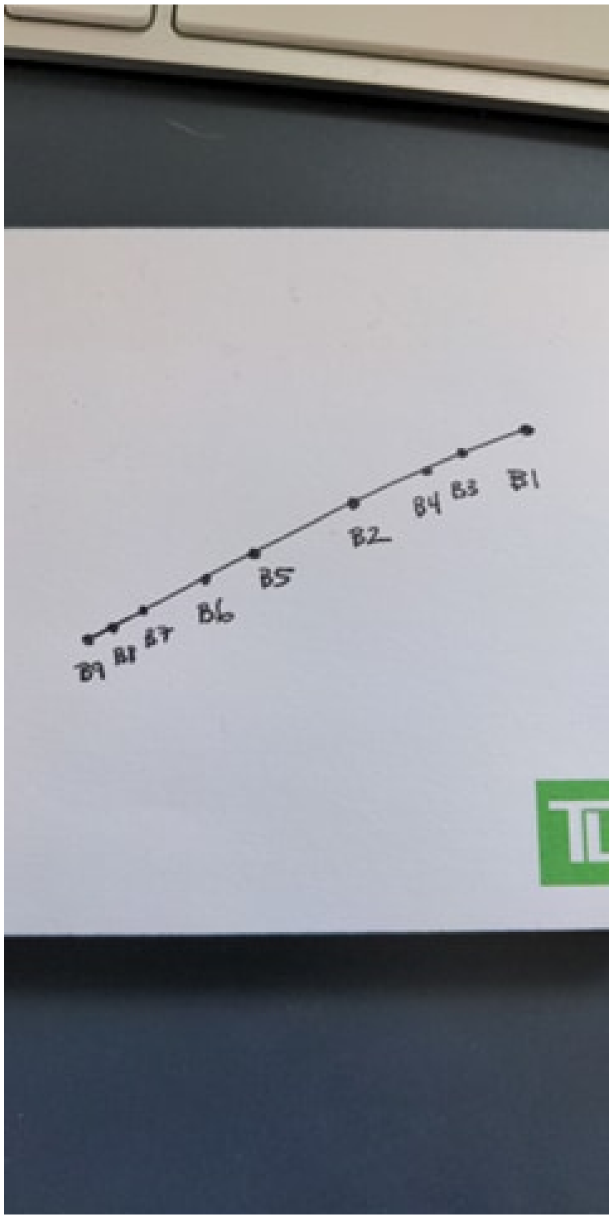
(a)

Clustering is a good idea here since we are not looking to predict a specific outcome. Instead we are trying to associate or group new birds with birds of the same type. We are trying to combine birds of similar characteristics based on a few recorded features.

(b)

- Remove outliers and impute missing values since KMeans cannot handle them.
- Normalize all variables so that euclidean distance isn't influenced in large part by a subset of the features.

(c)



(d)

Given the above-mentioned facts, I would use the centroid or average linkage. To assess the similarity or dissimilarity between groups, since they arrive in groups/clusters, we can take the average of all the birds of a given category and compare it to another groups average, essentially implementing distance between centroids as our main metric to build our hierarchical clustering diagram.

(e)

We are not given very much information and since we have few points and no guarantee that the data we are dealing with is hyper spherical, it is somewhat more appropriate to use hierarchical clustering.

(f)

We can use a combination of the following 2 metrics to assess the accuracy of our clustering.

- Cluster Cohesion: We measure the distance between points within a cluster and try to minimize this value.
- Cluster Separation: We measure the distance between clusters and try to maximize this value.

0.3 Cross Validation

Q3 10 points = $(1 + 2 + 3 + 1.5 + 1 + 1.5) $

Given below is a pair of plots generated while cross-validating a K-NN model trained on a dataset with various values of K using 12-fold CV and Leave-one-out (LOOCV) methods :

Answer the following questions:

(a) What is the motivation behind using cross-validation techniques such as LOOCV and k-fold CV over having a validation set?

(b) What could possibly explain the difference in the error curves in the two methods in the plots?

(c) Describe how similar (or) different the error curves generated on the same dataset would look like for another independent run of **each of** the methods *12-fold* and *LOOCV*, compared to the plots above. **Explain** the reason for your answer in each case.

(d) Under which circumstances would you would favour using LOOCV over k-fold CV?

(e) Based on the above plots, what is the best value for the hyperparameter K of the model? **Explain why.**

(f) What type of hyperparameter search do the above plots illustrate? **Explain**.

(a) The main motivations are that we can reduce the bias in our model and that we can utilize the entirety of our training set. Having a validation set limits the size of our training set and therefore prevents us from extracting all of the insight from our training set.

(b) The fact that LOOCV has access to a greater proportion of the training set per iteration could explain why overall LOOCV seems to be a much better performer than 12-fold CV. It will yield approximately unbiased estimates compared to the somewhat biased 12-fold CV estimates.

(c) Assuming we are dealing with independent estimates, the LOOCV curve will look similar to the one presented since it will have lower variance between models/iterations. We have lower variance since we're moving a single data point and so we'll have a very high rate of overlap between our training sets. This is not the case however in the case of KFold-CV. We can expect to see a quite different curve if we run it again.

(d) If we have a very small dataset, then using LOOCV would be preferable over KFold CV since we want to be able to use the most of the training set as we can for training.

(e) A K value of 5 seems appropriate as it yields the lower error in both LOOCV and KFold CV. It also happens to be a pretty standard/default value for KFold CV.

(f) These plots demonstrate a manual hyperparameter search. Here, we are analysing a single hyperparameter K for both CV methods. We are checking to see what error each value yields and combined with our understanding of the bias variance tradeoff, are looking to choose the optimal K value.

0.4 Inference and Bootstrapping

Q4 16 points = $(1.5 + 1.5 + 1.5 + 1.5 + 2 + 2 + 1.5 + 1.5 + 1.5 + 1.5)$

The dataset loaded in the next cell consists of data from a drug trial experiment.
* `subject_type` indicates 0 for if a subject is a *control* and 1 if taking *treatment*. * `daily_dosage` indicates the dosage of the drug in millilitres (mL) * `life_expectancy` show the projected age (year) upto which that the subject is expected to live.

Let the field `subject_type` in the dataset correspond to x_{type}, `daily_dosage` to x_{dosage} and `life_expectancy` to y.

(a) Now, consider the regression :

$$y = \beta_{dosage}x_{dosage} + \beta_{type}x_{type} + \beta_0 + \epsilon$$

Write your code to perform this regression and list the coefficient estimates $\hat{\beta}_{dosage}$, $\hat{\beta}_{type}$ and $\hat{\beta}_0$ that you obtained by running your code.

(b) Run a bootstrap of the dataset over 500 iterations, and collect the coefficients $\hat{\beta}_{dosage}$, $\hat{\beta}_{type}$ and $\hat{\beta}_0$ that you obtain in each iteration.

(c) Use the coefficient estimates that you collected in (b) and estimate the standard errors of all the 3 coefficients $S.E.(\hat{\beta}_{dosage})$, $S.E.(\hat{\beta}_{type})$ and $S.E.(\hat{\beta}_0)$.

(d) Plot a histogram to observe the distribution of each of the collected coefficient estimates. What do you observe?

(e) Provide a 95% confidence interval for each of the coefficient estimates. What does this interval mean?

(f) Generate the scatterplot for the points in the dataset with `daily_dosage` on the x-axis, `life_expectancy` on the y-axis and the points colored by the `subject_type` value (separate colors to indicate the types 0 and 1) with the collected 500 bootstrap sampled fits overlaid. To make the plot easier to read, reduce the transparency of the lines.

(g) Based on all the above, **explain** intuitively what is conveyed by the plot you generated in (f).

(h) Make a scatterplot of the bootstrapped coefficients, $\left(\beta^*_{type}, \beta^*_{dosage}\right)$ against one another. **Comment** on the overall distribution of these two coefficients **and** the nature of correlation between them.

(i) Similar to above, estimate the coefficient estimates $\hat{\beta}_{dosage}$ and $\hat{\beta}_0$ and the standard error of the dosage coefficient $S.E.(\hat{\beta}_{dosage})$ by bootstrapping the dataset over 500 iterations and fitting the dataset in the regression :

$$y = \beta_{dosage}x_{dosage} + \beta_0 + \epsilon$$

(j) Comparing the results in (i) and (c), what can you comment about the relationship of `daily_dosage` and `subject_type` with `life_expectancy`?

Bonus : (3 points) > (k) In a bootstrap of the dataset over 500 iterations similar to the above, perform an independent t-test with an $\alpha = 0.05$ significance level, using `scipy.stats.ttest_ind` on the `life_expectancy` of these two groups . Collect the p-values and plot the p-value histogram.

```
[14]: trials = pd.read_csv(path.format('drug-trials.csv'))
```

(a)

```
[15]: trials.head()
```

```
[15]:    subject_type  daily_dosage  life_expectancy
      0             0        22.762         63.39666
      1             0        22.762         80.14504
      2             1        16.036         82.31560
      3             1        23.830         84.20141
      4             0         3.073         57.28034
```

```
[28]: X = trials[['daily_dosage', 'subject_type']]
      y = trials['life_expectancy']
      lr = LinearRegression().fit(X, y)
```

```
[29]: lr.intercept_
```

```
[29]: 65.67943220081068
```

```
[30]: lr.coef_
```

```
[30]: array([0.27154927, 5.6681168 ])
```

B_dosage = 0.2715

B_type = 5.668

B_0 = 65.6794

(b)

```
[57]: coef_dosage = []
      coef_type = []
      coef_0 = []

      for i in range(500):
          # Resample from data with replacement then split into X and y.
          trials_boot = resample(trials, replace=True)
          X_boot = trials_boot[['daily_dosage', 'subject_type']]
          y_boot = trials_boot['life_expectancy']
```

```
    # Fit regressor and store coefs.
    lr = LinearRegression().fit(X_boot, y_boot)
    coef_dosage.append(lr.coef_[0])
    coef_type.append(lr.coef_[1])
    coef_0.append(lr.intercept_)
```

```
[141]: len(coef_0), len(coef_dosage), len(coef_type)
```

```
[141]: (500, 500, 500)
```

(c)

```
[59]: np.std(coef_dosage)
```

```
[59]: 0.11139904514756391
```

```
[60]: np.std(coef_type)
```

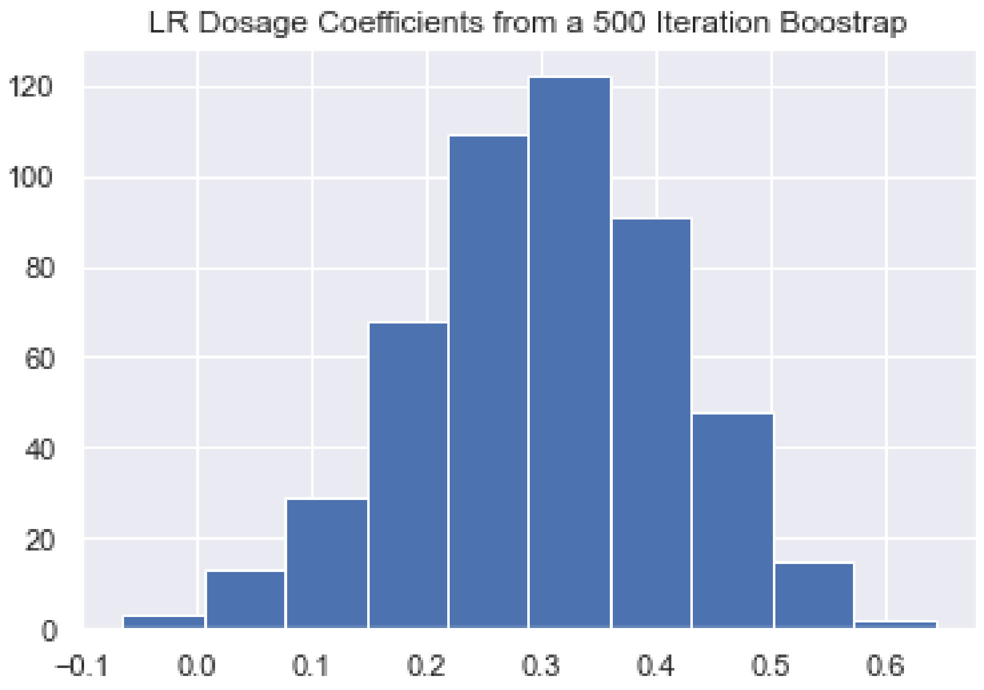
```
[60]: 1.8537206181000243
```

```
[61]: np.std(coef_0)
```

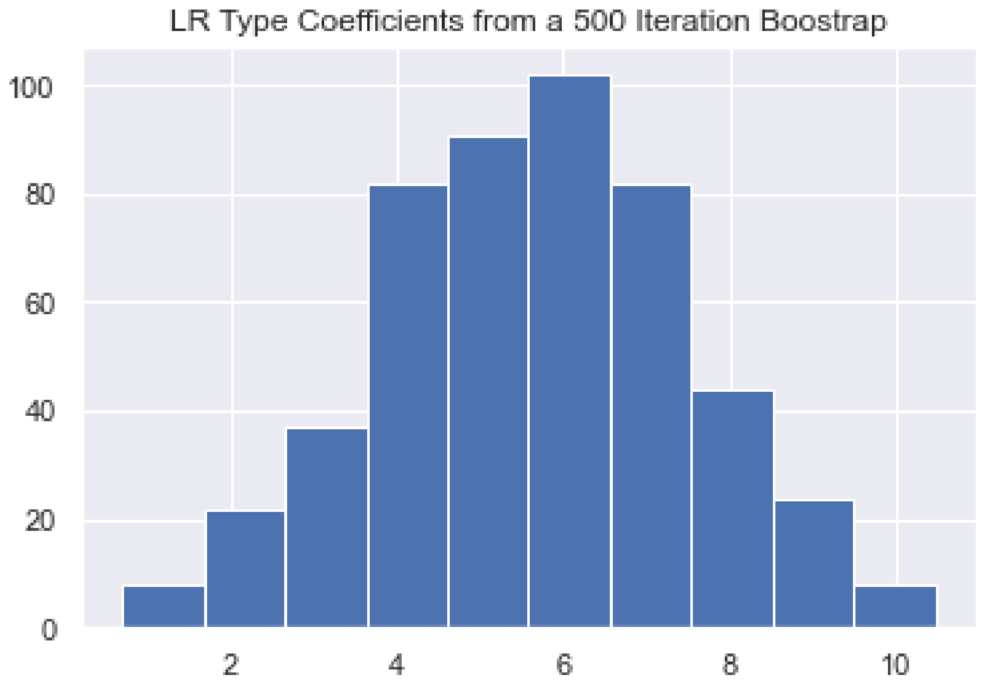
```
[61]: 2.3145047405328287
```

(d)

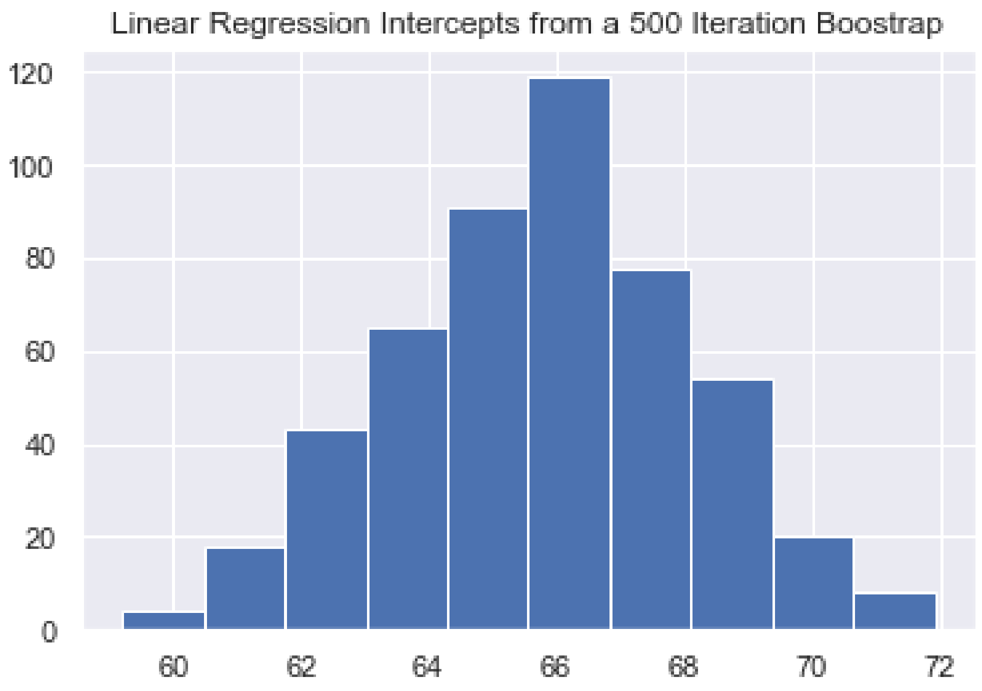
```
[143]: plt.title("LR Dosage Coefficients from a 500 Iteration Boostrap")
       plt.hist(coef_dosage);
```



```
[142]: plt.title("LR Type Coefficients from a 500 Iteration Boostrap")
       plt.hist(coef_type);
```



```
[70]: plt.title("Linear Regression Intercepts from a 500 Iteration Boostrap")
      plt.hist(coef_0);
```



Here, I observe that all of the coefficients seem normally distributed.

(e)

```
[71]: [np.mean(coef_dosage) - 1.96*np.std(coef_dosage), np.mean(coef_dosage) + 1.96*np.
       ↪std(coef_dosage)]
```

```
[71]: [0.04843414356615369, 0.4851184005446042]
```

```
[72]: [np.mean(coef_type) - 1.96*np.std(coef_type), np.mean(coef_type) + 1.96*np.
       ↪std(coef_type)]
```

```
[72]: [2.023685554532826, 9.290270377484921]
```

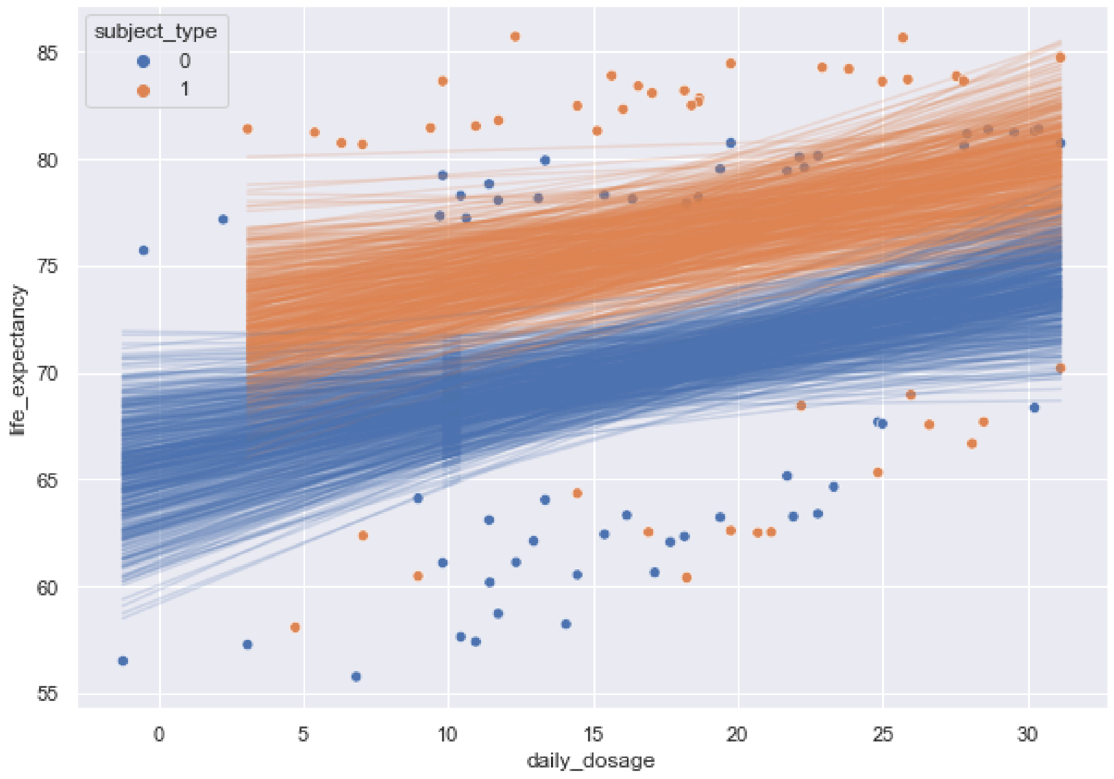
```
[73]: [np.mean(coef_0) - 1.96*np.std(coef_0), np.mean(coef_0) + 1.96*np.std(coef_0)]
```

```
[73]: [61.24290230798341, 70.3157608908721]
```

These intervals represent the range in which the values of each coefficient will fall into, in 95% of cases. It gives us a reasonable expectation of the coef values we should be getting.

(f)

```
[93]: plt.figure(figsize=(10,7))
      for i in range(500):
          l = sns.lineplot(x=trials['daily_dosage'],
                           y=coef_dosage[i]*trials['daily_dosage'] +␣
       ↪coef_type[i]*trials['subject_type'] + coef_0[i],
                           hue=trials['subject_type'],
                           alpha=0.2,
                           legend=False)
      sns.scatterplot(x=trials['daily_dosage'], y=trials['life_expectancy'],␣
       ↪hue=trials['subject_type']);
```

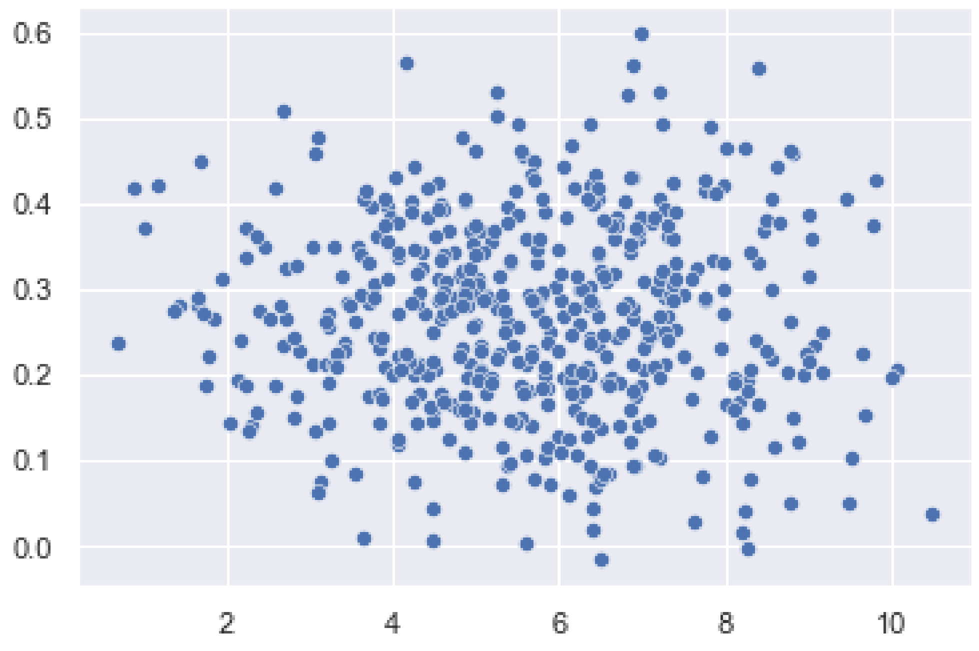


(g)

What we can infer intuitively from this graph is that, generally speaking, subject_type = 1 have a higher life expectancy than blue and coupled with a higher daily dosage, we maximize life expectancy, so daily dosage is positively correlated to life expectancy.

(h)

```
[97]: sns.scatterplot(x=coef_type, y=coef_dosage);
```



From this plot of coefficients we can clearly observe that our daily dosage and subject type coefficients are not correlated. Therefore, this tells us that our features are not correlated, which removes the worry of multicollinearity in our model.

(i)

```
[115]: coef_dosage = []
       coef_0 = []

       for i in range(500):
           # Resample from data with replacement then split into X and y.
           trials_boot = resample(trials, replace=True)
           X_boot = np.array(trials_boot[['daily_dosage']]).reshape((-1, 1))
           y_boot = trials_boot[['life_expectancy']]

           # Fit regressor and store coefs.
           lr = LinearRegression().fit(X_boot, y_boot)
           coef_dosage.append(lr.coef_[0][0])
           coef_0.append(lr.intercept_[0])
```

```
[146]: np.mean(coef_type)
```

```
[146]: 5.656977966008873
```

```
[118]: np.std(coef_dosage)
```

```
[118]: 0.11428382812749135
```

(j) We can confirm that both variables are positively correlated with life expectancy. For every unit of daily dosage, we have an average of 0.3 unit increase in life expectancy and for subject type 1 we have an average of 5.6 unit increase in life expectancy.

0.5 Feature Engineering

Q5 4 points = $(1.5 + 2.5)$

Given below is a cell that loads a dataset that contains features representing the body measurements of certain types of sharks in various regions in Canada.

Using programming, perform analyses using the following methods to identify the outlier samples in the dataset :

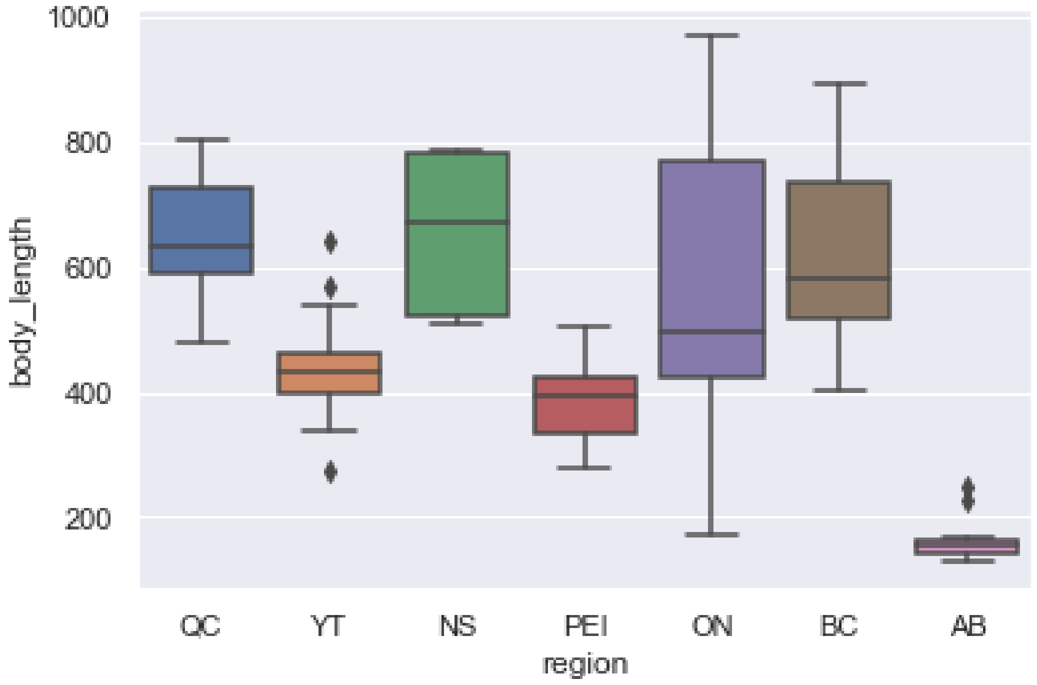
(a) For each *feature* among `body_length`, `fin_length` and `tail_length` in the dataset, use a box plot to visualize the feature values (along y-axis) grouped by `region` feature (show on x-axis). Identify the `region` and `feature` (other than `region`) that shows the highest number of outliers.

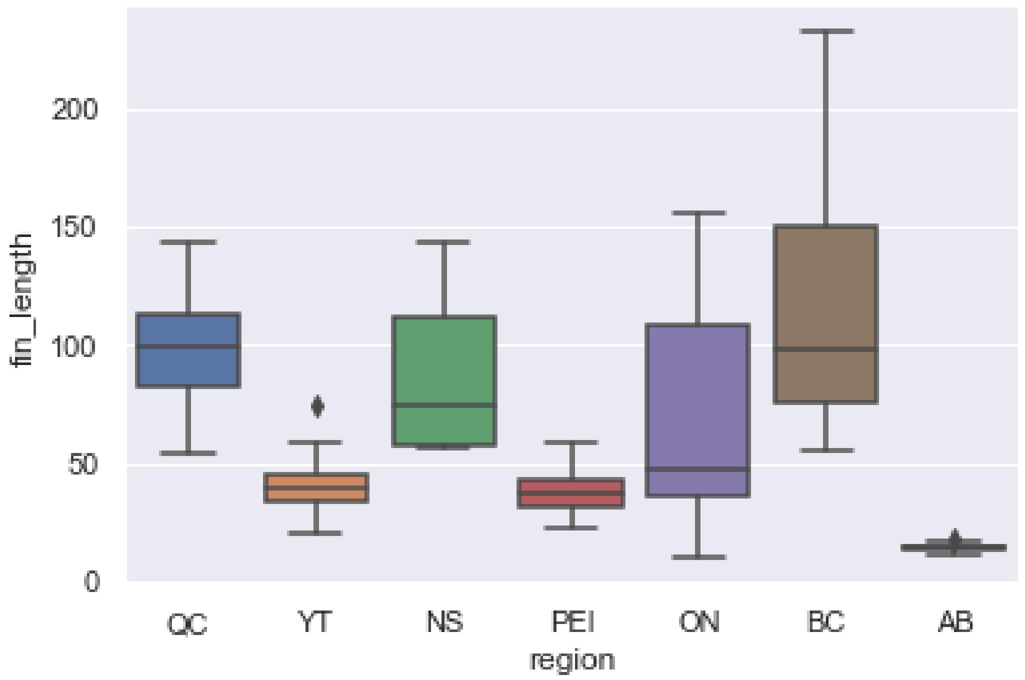
(b) For the `region` and the `feature` you chose in (a), use the Q3-Q1 Inter-Quartile Range (IQR) to identify and list the rows of the outliers from the dataframe.

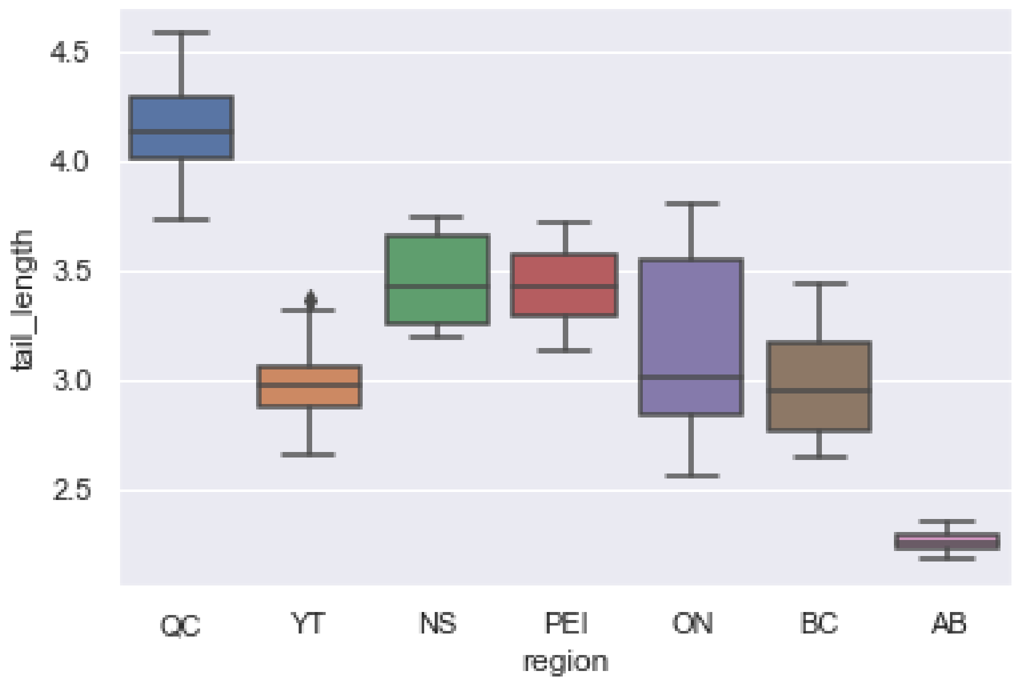
```
[149]: sharks = pd.read_csv(path.format('ca-sharks.csv'))
```

(a)

```
[150]: for f in ['body_length', 'fin_length', 'tail_length']:
           sns.boxplot(x='region', y=f, data=sharks)
           plt.show();
```


1000
800
600
400
200
body_length
QC
YT
NS
PEI
ON
BC
AB
region


200
150
100
50
0
fin_length
QC
YT
NS
PEI
ON
BC
AB
region



The body length feature shows the most number of outliers, namely in the YT region. (There are a small amount of outliers in AB region as well)

(b)

```
[161]: Q1 = np.percentile(sharks[sharks['region']=='YT']['body_length'], 25)
       Q3 = np.percentile(sharks[sharks['region']=='YT']['body_length'], 75)
       IQR = Q3-Q1
```

```
[162]: IQR
```

```
[162]: 63.73342500000001
```

```
[163]: outliers = [bl for bl in sharks[sharks['region']=='YT']['body_length'] if bl <=␣
        ↪Q1 - 1.5*IQR or bl >= Q3 + 1.5*IQR]
```

```
[164]: outlier_subset = sharks[(sharks['region']=='YT') & (sharks['body_length'].
        ↪isin(outliers))]
```

```
[165]: outlier_subset
```

```
[165]:     id region  fin_length  body_length  tail_length
       35  36     YT        20.2     272.3920     2.653751
       53  54     YT        57.8     570.5584     3.253912
       54  55     YT        74.0     639.7450     3.365487
```

Q6 8 points = $(1+3+2+2)$

For the questions below, answer briefly by inspecting the dataset below. (There is no need to use any programming) :

This is a representative subset of a collected dataset with information about used buses across three Canadian cities. A model needs to be fit to predict the selling price `Price($)` of a bus. `--` indicates that the information is not available.

(a) List the features that you would remove from the dataset before using it for model fitting. Give valid **reasons** for your answer.

(b) List the features that need to be encoded in this dataset and outline which encoding schemes your would use in each case. Give valid **reasons** for your answer.

(c) What type of an imputation scheme would make sense for the missing values in each of the fields `Year` and `Mileage`? Give valid **reasons** for your answer.

(d) You propose to use the simplest sparsity-based method to select the best features among those given in the dataset. In just two lines, describe the high-level procedure to do this.

(a)

I would remove Vehicle model as it is information that is likely to suffer from high dimensionality and will add noise to our predictions and increase the runtime of our model training. I would also remove City since it will likely be highly correlated with Province and multicollinearity will occur.

(b)

Once we remove the features mentioned above, we will one hot encode Province and Category and we will label encode Certification. We will one-hot encode Prov and Cat since these categorical variables cannot be ordered in any meaningful way relative to the target variable whereas certification can be ordered. Price seems to be positively correlated with the presence of certification in buses (as observed from the small subset of data).

(c)

Year: We will impute by the mode year per Category. We can't use mean or median since this will likely produce decimal values. Tourism buses will probably want to update their buses frequently and so they will have more recent buses whereas factory buses are less concerned with the appearance and functions of their buses and more concerned with saving money, making a category grouping somewhat appropriate.

Mileage: We will impute by group average mean, where we group the observations by year. This is reasonable since newer cars/buses are generally more fuel efficient.

(d)

We will implement Lasso feature selection. Lasso will set some feature coefficients to 0 to reduce the number of correlated variables, then it will find the coefficients that minimize the objective (penalized by a regularization term lambda)

[ ]: