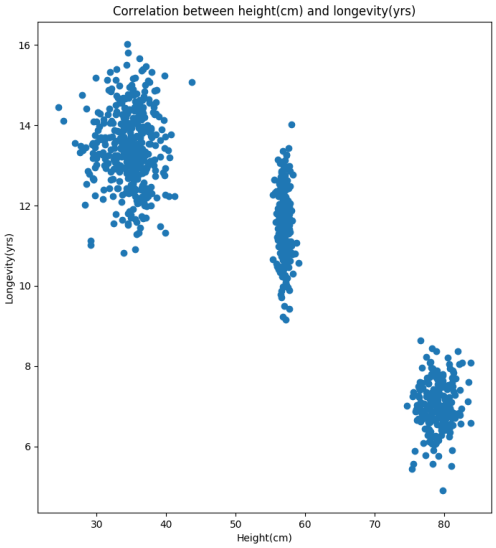
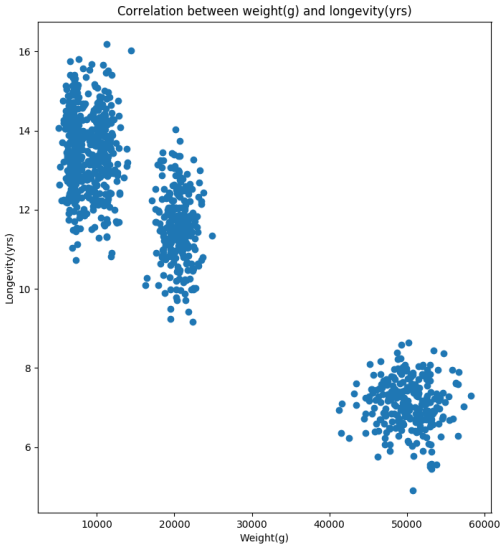
Task 2 – Regression

Data set : 7 features(2 numerical, 5 strings) and one label(numerical - Longevity)
- normalized the 2 numerical values(Weight and Height)
- filled the NaN cells with the mean of that column
- one-hot encoded Energy Level, Attention Needs, Coat Length and Sex
- dropper the Owner Name column so we'll have 6 features
- split the data set in:training set(75%) and test set(25%)

Output variables :
- values between 4.909 and 16.1829 with a mean equal to 11.306
- at first I tried to see the correlation between height/weight and the longevity





- higher the weight/height,lower the longevity.These 2 features seemed the most related to the longevity.The correlation plot for both features are alike.This is quite natural because bigger the height,bigger the weight.
- I started with the assumption that these 2 features will be the most important and a model having just these two will be promising.

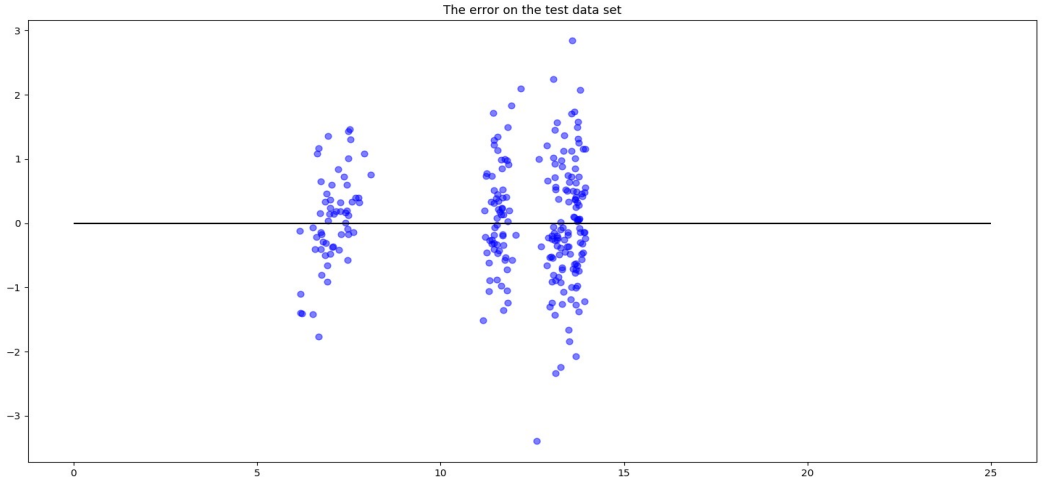
Starting the learn with Linear Regression:

- at first I used default Sklearn Linear Regression:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
         normalize=False)
```
with all the features
- the coefficients estimated showed again the strong correlation between weight/height and longevity,but biased more on the weight-longevity.Next I tried to drop columns like energy level,attention,coat length,sex while the weight was still there.Given these results(and the correlation plots from above),I was determined to use only weight and height.

	features	estimatedCoefficients
0	Weight(g)	-7.711841
1	Height(cm)	-0.478157
2	energy_high	0.031446
3	energy_low	-0.060209
4	energy_med	0.028763
5	attention_high	0.015356
6	attention_med	-0.015356
7	coat_long	-0.014348
8	coat_med	-0.025753
9	coat_short	0.040101
10	sex_female	-0.021971
11	sex_male	0.021971

- the results for the current aproach are:

```
Linear regression cross validation score : 0.8843457344043735
Linear regression accuracy : 0.8979960754295238
Mean Absolute Error : 0.6819252258453204
Mean Squared Error : 0.7670352797455849
Root Mean Squared Error : 0.005389031884849654
```



- the plot above shows the difference between the prediction and the real label.Most of the data was predicted with an error lower than 1,but there are also some extremely bad predictions(2-3 years error).
- the *Linear regression accuracy* was r squared score which I found being the most used in practice,despite the fact that it could hide an overfitting.On the other hand the cross validation score was near to the r squared one so it seems that the model isn't overfitting.

Optimisations:

- setting the test set size to 33% increased a little the r squared score

```
Linear regression cross validation score : 0.8812972094382132
Linear regression accuracy : 0.9001490271257335
Mean Absolute Error : 0.6814321683241231
Mean Squared Error : 0.7642194596290811
Root Mean Squared Error : 0.005347425280469649
```

- adding regularization.Ridge has a little increase in performance in the most of the cases

```
Ridge(alpha=0.1, copy_X=True, fit_intercept=True, max_iter=1000,
   normalize=False, random_state=None, solver='sag', tol=0.001)
Linear regression cross validation score : 0.8812802750476237
Linear regression accuracy : 0.9001674773172785
Mean Absolute Error : 0.6818669614528828
Mean Squared Error : 0.7640782492331605
Root Mean Squared Error : 0.005347223453776762
```

- I also tried other combinations,but with no result: ElasticNet(variations on l1_ratio),Lars, Lars Lasso etc.

Final model: the data set is split in 66% training and 33% test and has only Weight and Height as features;used the Ridge Regression with *sag* solver and alpha=0.1.The scores varies a few from the values from above.The cross validation score is obtained for cv=5.

Starting the learn with KNN Regression:

- used the same data set as above with a default KneighborsRegressor.The results are shown bellow:

```
KNN cross validation score : 0.8711585071114486
KNN accuracy : 0.8775960704609334
Mean Absolute Error : 0.7515773939393938
Mean Squared Error : 0.9368307808739395
Root Mean Squared Error : 0.006107125828838181
```

Conclusions: the *Linear Regression* would be a better choice.

Task 1 – Classification

Data set : 7 features(2 numerical, 5 strings) and one label(numerical - Longevity)

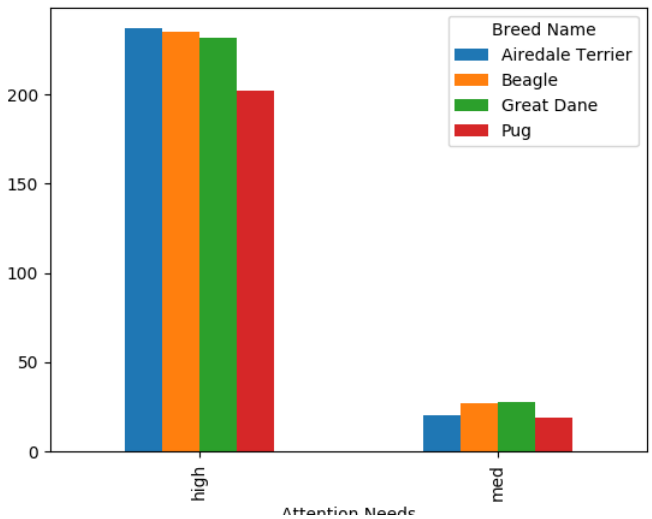
- normalized the 2 numerical values(Weight and Height)
- filled the NaN cells with the mean of that column
- one-hot encoded Energy Level, Attention Needs, Coat Length and Sex
- dropper the Owner Name column so we'll have 6 features
- split the data set in:training set(75%) and test set(25%)

Output variables:

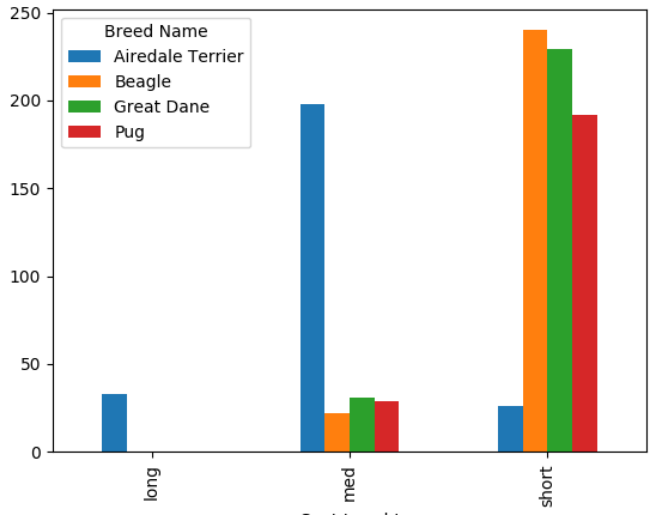
```
Beagle              262
Great Dane          260
Airedale Terrier    257
Pug                 221
```

- there are 4 classes distributed almost equaly

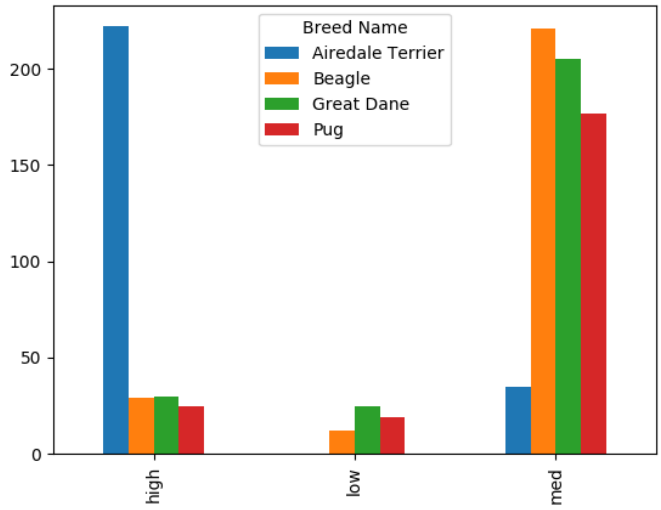
Features vizualization:



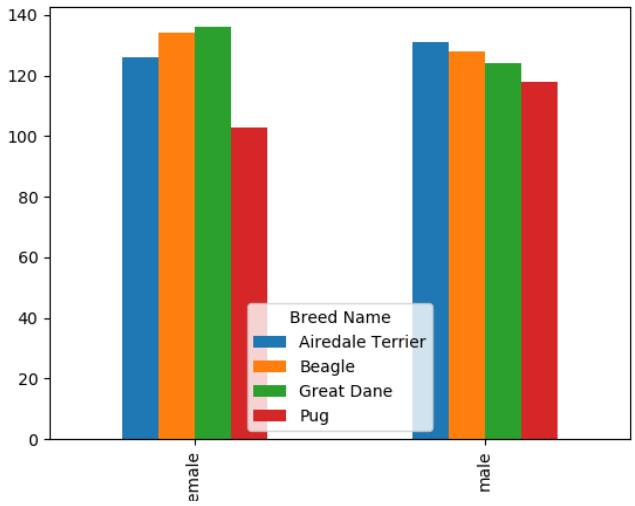
Attention needs – bad predictor



Coat length – bad predictor



Energy level – bad predictor



Gender – bad predictor

-only weight and height would be good predictors,so we'll drop other features.

Starting the learn with Logistic Regression:

- started with default Logistic Regression using all the features

```
Logistic regression cross validation score : 0.7692542779679096
Logistic regression accuracy : 0.816
Confusion matrix :
[[65  0  0  0]
 [11 56  0  0]
 [ 0  0 60  0]
 [13 22  0 23]]
```

- the predictions for the 4th class are quite bad,so it's the time to play a little with the LogisticRegression's parameters.
- I set multi_class='multinomial', solver='lbfgs', penalty='l2' and tol=0.01 and the results were better. The 4th class has less errors and the accuracy went to 90%.

```
Logistic regression cross validation score : 0.8959784286709039
Logistic regression accuracy : 0.9
Confusion matrix :
[[65  0  0  0]
 [11 50  0  6]
 [ 0  0 60  0]
 [ 5  3  0 50]]
```

- after more experimenting I reached this accuracy using multi_class='multinomial', solver='saga', penalt='l1' and tol=0.01 with a data set ratio of 75-15.The errors appear when trying to predict the breed for the 2nd class(Beagle) which is sometimes confused with the 4th one(Pug) probably due to their little difference between height and weight(compared to the other 2 classes which respresent big dogs).

```
Logistic regression cross validation score : 0.9765182564253772
Logistic regression accuracy : 0.9666666666666667
Confusion matrix :
[[38  0  0  0]
 [ 0 33  0  5]
 [ 0  0 35  0]
 [ 0  0  0 39]]
```

Starting the learn with KNN Classifier:

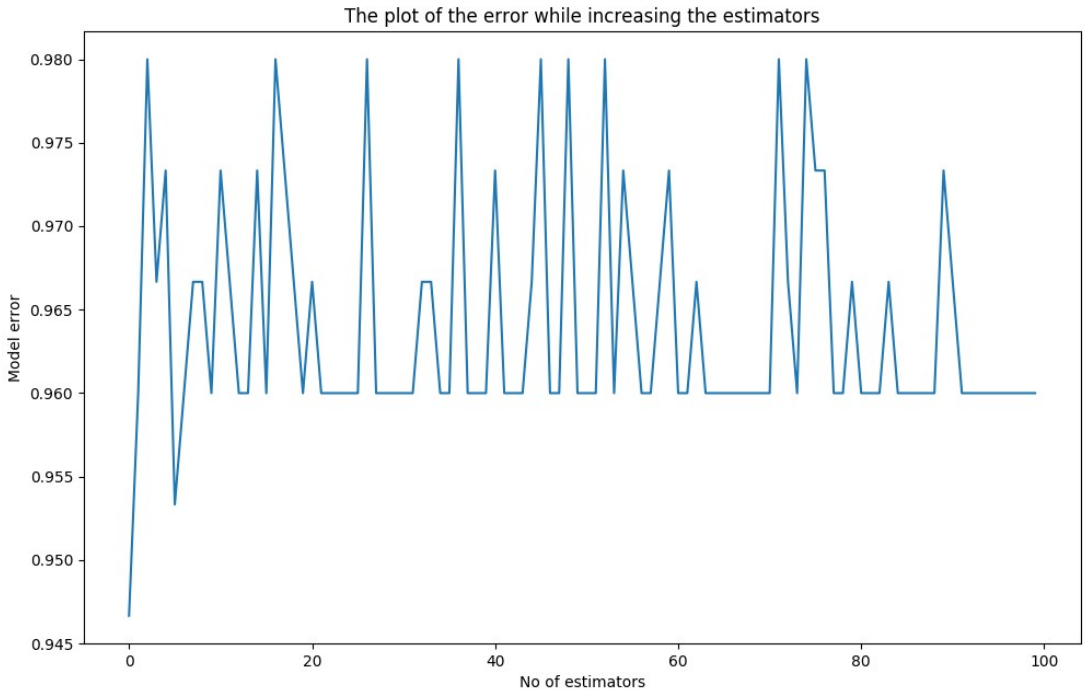
- used the same data set as above with a default *KneighborsClassifier*. I got a good result from the first try so I didn't continue to change the hyperparameters.The results are shown below:

```
KNN cross validation score : 0.9764973708781758
KNN accuracy : 0.98
Confusion matrix :
[[38  0  0  0]
 [ 0 35  0  3]
 [ 0  0 35  0]
 [ 0  0  0 39]]
```

Conclusions: KNN Classifier would be a better choice.

Starting the learning with Random Forests:

- I started with the plot of the errors produced by the model while the n_estimator parameter was increased.The plot changed every time,but I noticed that usually an estimator chosen between 10-100 would usually perform better than other choices.



- the Gini criterion performed better than the Entropy
- the best accuracy looks like this:

```
Random forests cross validation score : 0.96
Random forests accuracy : 0.98
Confusion matrix :
[[38  0  0  0]
 [ 0 35  0  3]
 [ 0  0 35  0]
 [ 0  0  0 39]]


                precision    recall  f1-score   support

Airedale Terrier       1.00      1.00      1.00        38
         Beagle       0.92      1.00      0.96        35
     Great Dane       1.00      1.00      1.00        35
            Pug       1.00      0.93      0.96        42

      micro avg       0.98      0.98      0.98       150
      macro avg       0.98      0.98      0.98       150
   weighted avg       0.98      0.98      0.98       150
```

- I’ve used n_estimators=10,criterion=’gini’ and oob_score=True to get out of bag validation(which I consider an equivalent of cross validation)

Conclusions: there are small differences between the models presented above,but I think that the *Logistic Regression* would be the best choice because it isn’t prone to overfitting(compared to the other two models).