# IAML – INFR10069 (LEVEL 10): Assignment #1

s1853813

## Question 1 : (22 total points) Linear Regression

**In this question we will fit linear regression models to data.**

(a) (3 points) Describe the main properties of the data, focusing on the size, data ranges, and data types.

The dataset contains 50 instances and 2 attributes(`exam_score` and `revision_time`.
The data range for `exam_score` is between 2.723000 and 48.011000.
The data range for `revision_time` is between 14.731000 and 94.945000.
The data type for `exam_score` and `revision_time` are both float64.

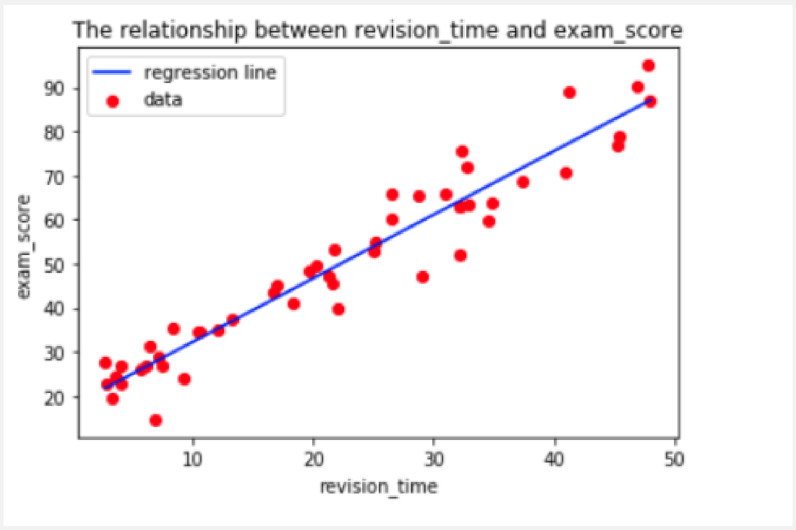
*Question 1 continued on next page. . .*

(b) (3 points) Fit a linear model to the data so that we can predict `exam_score` from `revision_time`. Report the estimated model parameters **w**. Describe what the parameters represent for this 1D data. For this part, you should use the sklearn implementation of Linear Regression.
*Hint: By default in sklearn* `fit_intercept = True`*. Instead, set* `fit_intercept = False` *and pre-pend* 1 *to each value of* $x_i$ *yourself to create* $\boldsymbol{\phi}(x_i) = [1, x_i]$.

`w` = [17.8977, 1.4411]
`w` is a 2x1 array and w[0] means the y-intercept for the regression line and the w[1] is the best fit slope of the linear regression line that makes the minimum error among the data points. In this case, the y-intercept is 17.8977 and the slope is 1.4411.

*Question 1 continued on next page...*

(c) (3 points) Display the fitted linear model and the input data on the same plot.



*Question 1 continued on next page...*

(d) (3 points) Instead of using sklearn, implement the closed-form solution for fitting a linear regression model yourself using numpy array operations. Report your code in the answer box. It should only take a few lines (i.e. <5).
*Hint: Only report the relevant lines for estimating* **w** *e.g. we do not need to see the data loading code. You can write the code in the answer box directly or paste in an image of it.*

```
w_estimiate = np.dot(np.linalg.inv(np.dot(X.T, X)), np.dot(X.T, y))
```

*Question 1 continued on next page. . .*

(e) (3 points) Mean Squared Error (MSE) is a common metric used for evaluating the performance of regression models. Write out the expression for MSE and list one of its limitations.
*Hint: For notation, you can use y for the ground truth quantity and $\hat{y}$ (`$\hat{y}$` in latex) in place of the model prediction.*

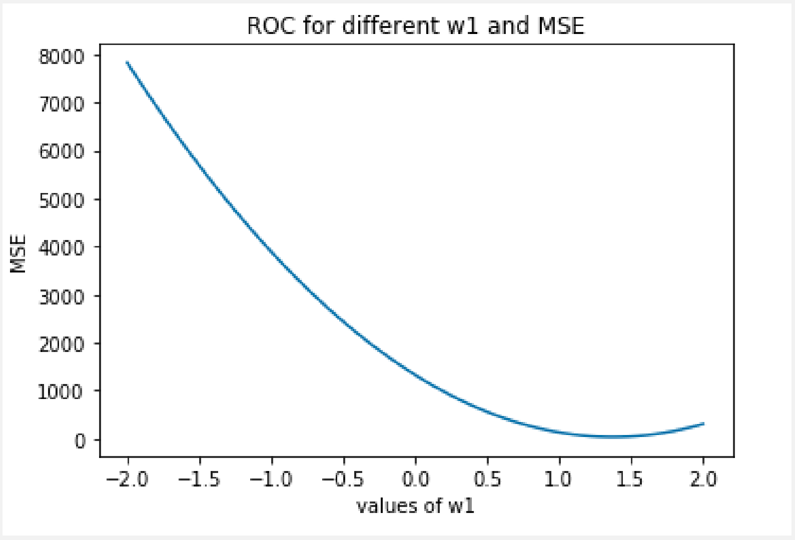
MSE $=\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y})^2$
One of the limitations of MSE is that MSE is that it is very sensitive to the outliers. Even one outlier in the test data with increase the MSE significant large which reduce the accuracy of the well fit regression line. It cannot tell the difference between a outlier is put into well fit regression line and no outlier is put into regression line with less fit since at the end the MSE formula is taking the sum of the errors.

*Question 1 continued on next page...*

(f) (3 points) Our next step will be to evaluate the performance of the fitted models using Mean Squared Error (MSE).
Report the MSE of the data in `regression_part1.csv` for your prediction of `exam_score`. You should report the MSE for the linear model fitted using sklearn and the model resulting from your closed-form solution. Comment on any differences in their performance.

The MSE of the predictions of using the sklearn is 30.985472614541294 while the MSE of the predictions of using the closed-form solution above is 30.98547261454129. There is not much a difference since the w in training the model in sklearn and the closed-form solution are same. We using the same X with same w to predict $\hat{y}$. Therefore, the difference is minimal like the sklearn one has a more precise calculations of extra digit of decimal the end.

(g) (4 points) Assume that the optimal value of $w_0$ is 20, it is not but let's assume so for now. Create a plot where you vary $w_1$ from $-2$ to $+2$ on the horizontal axis, and report the Mean Squared Error on the vertical axis for each setting of $\mathbf{w} = [w_0, w_1]$ across the dataset. Describe the resulting plot. Where is its minimum? Is this value to be expected? *Hint: You can try 100 values of $w_1$ i.e.* `w1 = np.linspace(-2,2, 100)`.

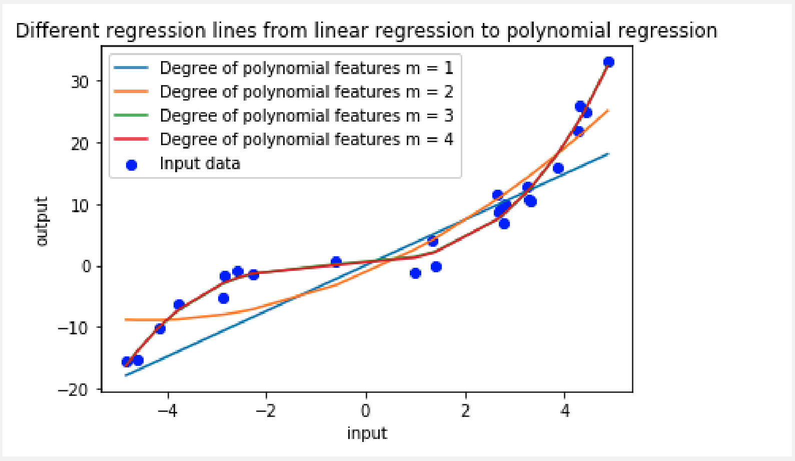


The plot shows that the MSE begins to drop with a higher value of w1. MSE starts to rise when w1 has the minimum point 1.3535. It quite make sense that the MSE drops when the value of w1 increases. Since w1 is the slope of the regression line, the original plot of the dataset shows a positive correlation between `revision_time` and `exam_score`. Therefore, it is w1 is positive has a low error than the negative w1. After the minimum point in 1.3535, the MSE begins to rise because the regression line performs badly with catching the linear relationship of the data points. And 1.3535 is quite close with the w1 sklearn and our closed-form solution. Therefore, I believe 1.3535 is a reasonable point for the minimum.

## Question 2 : (18 total points) Nonlinear Regression

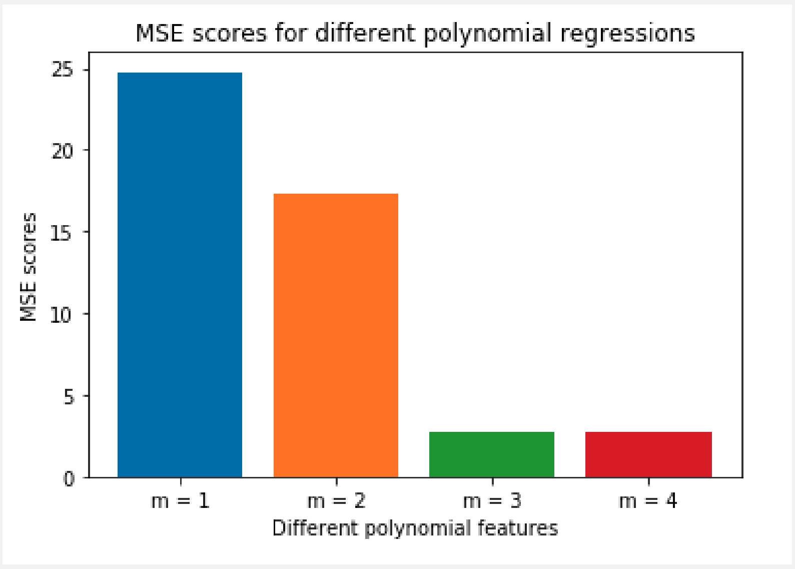
**In this question we will tackle regression using basis functions.**

(a) (5 points) Fit four different polynomial regression models to the data by varying the degree of polynomial features used i.e. $M = 1$ to 4. For example, $M = 3$ means that $\boldsymbol{\phi}(x_i) = [1, x_i, x_i^2, x_i^3]$. Plot the resulting models on the same plot and also include the input data.
*Hint: You can again use the sklearn implementation of Linear Regression and you can also use PolynomialFeatures to generate the polynomial features. Again, set `fit_intercept = False`.*



*Question 2 continued on next page...*

(b) (3 points) Create a bar plot where you display the Mean Squared Error of each of the four different polynomial regression models from the previous question.

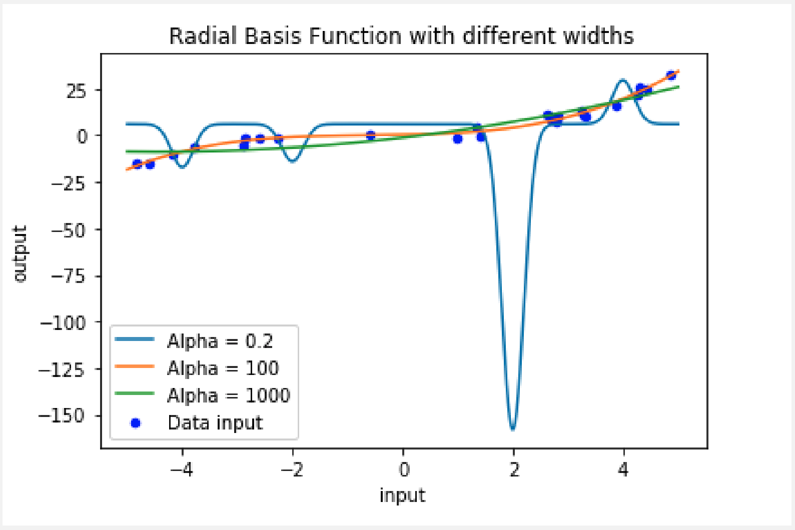


(c) (4 points) Comment on the fit and Mean Squared Error values of the $M = 3$ and $M = 4$ polynomial regression models. Do they result in the same or different performance? Based on these results, which model would you choose?

To do the fitting to the polynomial regression with Polynomial Feature m = 3 and m = 4. We transformed the original X which has shape (25,1) to higher dimension which has (25,4) and (25,5) for m = 3 and m = 4 respectively. Therefore, we can train our model and see how the regressions preform by looking at the Mean Squared Error. In terms of MSE values they are quite low and similar(2.745 and 2.7389 for m = 3 and m = 4 respectively). We can conclude that their performance are similar but m = 4 is slightly better which means the regression for m = 4 fits better with the data. Given we have a small amount of data and did not have test data or outliers, I would choose the polynomial regression model with m = 3 for less computation.

*Question 2 continued on next page...*

(d) (6 points) Instead of using polynomial basis functions, in this final part we will use another type of basis function - radial basis functions (RBF). Specifically, we will define $\boldsymbol{\phi}(x_i) = [1, rbf(x_i; c_1, \alpha), rbf(x_i; c_2, \alpha), rbf(x_i; c_3, \alpha), rbf(x_i; c_4, \alpha)]$, where $rbf(x; c, \alpha) = \exp(-0.5(x-c)^2/\alpha^2)$ is an RBF kernel with center $c$ and width $\alpha$. Note that in this example, we are using the same width $\alpha$ for each RBF, but different centers for each.
Let $c_1 = -4.0$, $c_2 = -2.0$, $c_3 = 2.0$, and $c_4 = 4.0$ and plot the resulting nonlinear predictions using the `regression_part2.csv` dataset for $\alpha \in \{0.2, 100, 1000\}$. You can plot all three results on the same figure. Comment on the impact of larger or smaller values of $\alpha$.



As in the plot, the smaller alpha value gives a overfit curve while higher alpha will have a smoother curve.

## Question 3 : (26 total points) Decision Trees

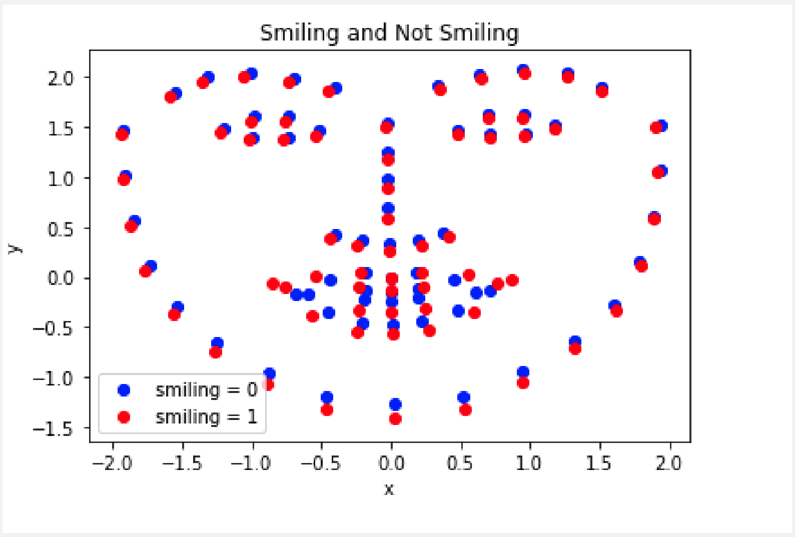
**In this question we will train a classifier to predict if a person is smiling or not.**

(a) (4 points) Load the data, taking care to separate the target binary class label we want to predict, `smiling`, from the input attributes. Summarise the main properties of both the training and test splits.

The training data contains 4800 rows and test data contains 1200 rows. Each instance (a face)has 137 attributes which are 68 x, y coordinates plus one classifier label. The label 1 means it is a smiling face and vice versa. The data type for the coordinates are float64 and the label is int64 . In the training set, there are 2335 smiling face and 2465 non-smiling face. Meanwhile, there are 592 smiling face and 608 non-smiling face in test set.

*Question 3 continued on next page. . .*

(b) (4 points) Even though the input attributes are high dimensional, they actually consist of a set of 2D coordinates representing points on the faces of each person in the dataset. Create a scatter plot of the average location for each 2D coordinate. One for (i) smiling and (ii) one not smiling faces. For instance, in the case of smiling faces, you would average each of the rows where `smiling = 1`. You can plot both on the same figure, but use different colors for each of the two cases. Comment on any difference you notice between the two sets of points.
*Hint: Your plot should contain two faces.*



The obvious difference between two sets of points are the points around the middle area . The points with smiling has wider points on x-axis around the middle([-1, 1]), while non-smiling lies between [-0.75, 0.75]. And the points around the middle for are more concentrated and the smiling ones spread wider. Therefore, we might conclude the way to conclude it is a smiling face is to look at the middle part of plot (mouth) as smiling will have a wider spread of data in the middle (mouth).

(c) (2 points) There are different measures that can be used in decision trees when evaluating the quality of a split. What measure of purity at a node does the DecisionTreeClassifier in sklearn use for classification by default? What is the advantage, if any, of using this measure compared to entropy?

By default, the DecisionTreeclassifier in sklearn uses Gini Impurity for the criterion in the DecisionTreeClassifier. The formula for Gini Impurity is 1 - $\Sigma_{i=1}^{n} p^2(ci)$ where p(ci) is the probability of ci in a node.
The advantage of using Gini Impurity over Entropy is that it does not have log in the equation which can require less computation and the results are basically the same.

*Question 3 continued on next page. . .*

(d) (3 points) One of the hyper-parameters of a decision tree classifier is the maximum depth of the tree. What impact does smaller or larger values of this parameter have? Give one potential problem for small values and two for large values.

The impact of having a small value for max depth would be the decision tree might not able to capture the pattern for the data. In another word, the tree stops when the impurity is still high and the predictions would be inaccurate. For high value for max depth, one of the impacts would be overfitting. High depth makes the splitting perfectly in training data while it behaviour poorly in test data since is not exactly like training data. Secondly, huge memory is used to store and to calculate for the tree. Accuracy drops after certain depth of a tree. Therefore, resources are wasted.

*Question 3 continued on next page...*

(e) (6 points) Train three different decision tree classifiers with a maximum depth of 2, 8, and 20 respectively. Report the maximum depth, the training accuracy (in %), and the test accuracy (in %) for each of the three trees. Comment on which model is best and why it is best.
*Hint: Set* `random_state = 2001` *and use the* `predict()` *method of the DecisionTreeClassifier so that you do not need to set a threshold on the output predictions. You can set the maximum depth of the decision tree using the* `max_depth` *hyper-parameter.*

Results are presented in the table below.

| Depth value | Train Accuracy | Test Accuracy |
|---|---|---|
| 2 | 79.48% | 78.17% |
| 8 | 93.35% | 84.08% |
| 20 | 100% | 81.58% |

We want the tree can predict well in the test accuracy and while the model with reasonable depth to capture the pattern in the training data and avoid overfitting. Among the three depths, the decision tree seems perform better with depth = 8 since the train accuracy and test accuracy are high and the difference is between 10%. We avoid depth = 20 because the train accuracy is 100% (overfitting) and test accuracy did not perform as well as depth = 8. Therefore, we can have a less computation to get a reasonable train accuracy and the best test accuracy among 3 with depth = 8.

*Question 3 continued on next page…*

(f) (5 points) Report the names of the top three most important attributes, in order of importance, according to the Gini importance from DecisionTreeClassifier. Does the one with the highest importance make sense in the context of this classification task?
*Hint: Use the trained model with* `max_depth = 8` *and again set* `random_state = 2001`*.*

According to DecisionTreeClassifier, the top 3 highest importance are 'x50', 'y48', 'y29' with corresponding importance vales (0.3304, 0.08996 and 0.08831). 'x50' is make sense since 'x50' around the middle part(mouth/lip). As for this classification task, the data points around the mouth play a significant role to classify whether if it is smiling.

*Question 3 continued on next page...*

(g) (2 points) Are there any limitations of the current choice of input attributes used i.e. 2D point locations? If so, name one.

There are some obvious limitations with choice of input attributes since they only allow the model to one position of the front face. However, the model would not work if the input that has a smaller face than the usual input, a face is turned to left or right a bit or even a rotated face.

## Question 4 : (14 total points) Evaluating Binary Classifiers

**In this question we will perform performance evaluation of binary classifiers.**

(a) (4 points) Report the classification accuracy (in %) for each of the four different models using the `gt` attribute as the ground truth class labels. Use a threshold of >= 0.5 to convert the continuous classifier outputs into binary predictions. Which model is the best according to this metric? What, if any, are the limitations of the above method for computing accuracy and how would you improve it without changing the metric used?

Results are presented in the table below.

| Model | Accuracy |
|---|---|
| alg_1 | 61.60% |
| alg_2 | 55.00% |
| alg_3 | 32.10% |
| alg_4 | 32.90% |

It is clear that alg_1 is the best among the algorithms. However, when we have unbalanced class, this metric would be deceiving. One of way to improve the accuracy is to use over-sampling to increase the size of the rare samples and the accuracy would be more representative.

*Question 4 continued on next page...*

(b) (4 points) Instead of using classification accuracy, report the Area Under the ROC Curve (AUC) for each model. Does the model with the best AUC also have the best accuracy? If not, why not?
*Hint: You can use the* roc_auc_score *function from sklearn.*

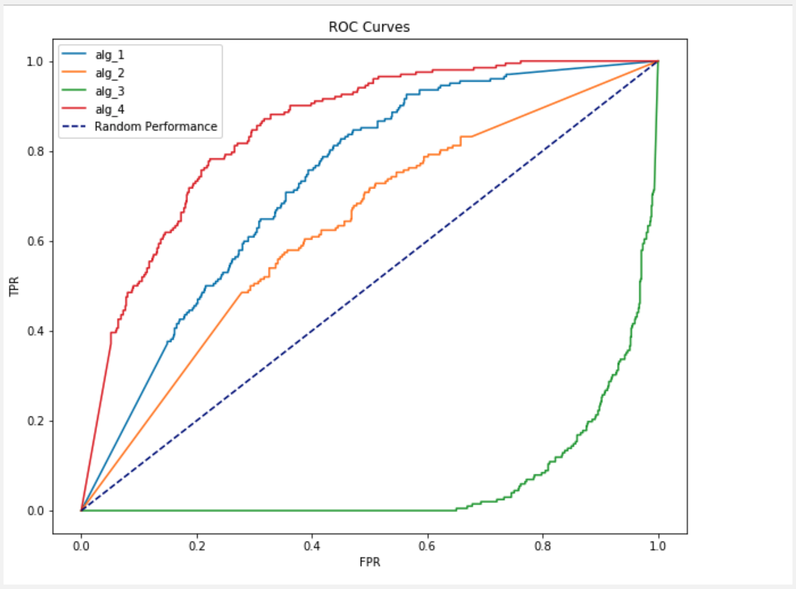
Results are presented in the table below.

| Model | AUC |
|---|---|
| alg_1 | 0.73 |
| alg_2 | 0.63 |
| alg_3 | 0.06 |
| alg_4 | 0.84 |

No, the best accuracy with threshold is alg_1. However, the AUC of alg_4 outperforms alg_1 by 0.11. Since the threshold we used in the beginning is 0.5, AUC computes the sum of the accuracies with all the threshold. The model with higher the AUC performs better in seperating two classes.

*Question 4 continued on next page. . .*

(c) (6 points) Plot ROC curves for each of the four models on the same plot. Comment on the ROC curve for `alg_3`? Is there anything that can be done to improve the performance of `alg_3` without having to retrain the model?
*Hint: You can use the roc_curve function from sklearn.*



We can see from the graph that alg_3 (green line) is entirely different from other lines and the performance is quite terrible. The performance of FPR is always larger or equal to FPR. In other words, the proportion of the gf that are incorrectly classified as 0 is greater than the proportion of correctly classified as 0. One simple way to improve of the performance is to reverse the predictions in alg_3. By flipped the predictions from 1 to 0 and 0 to 1. The performance would improve significantly we flipped the proportion of miss classified to opposite prediction so that we have higher possibility to get the predictions correct.