IAML – INFR10069 (LEVEL 10): Assignment #1

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 has 50 datapoints, with 2 attributes each: revision_time (by which it is sorted) and exam_score, both of which take continuous values. Revision_time has a mean of 22.22002, a standard deviation of 13.98611, and max and min values of 48.01100 and 2.72300. Exam_score has a mean of 49.91986, a standard deviation of 20.92559, and max and min values of 94.94500 and 14.73100 respectively.
A distplot of both attribute suggests that they may be normally distributed, although it is hard to tell exactly due to the small size of the dataset

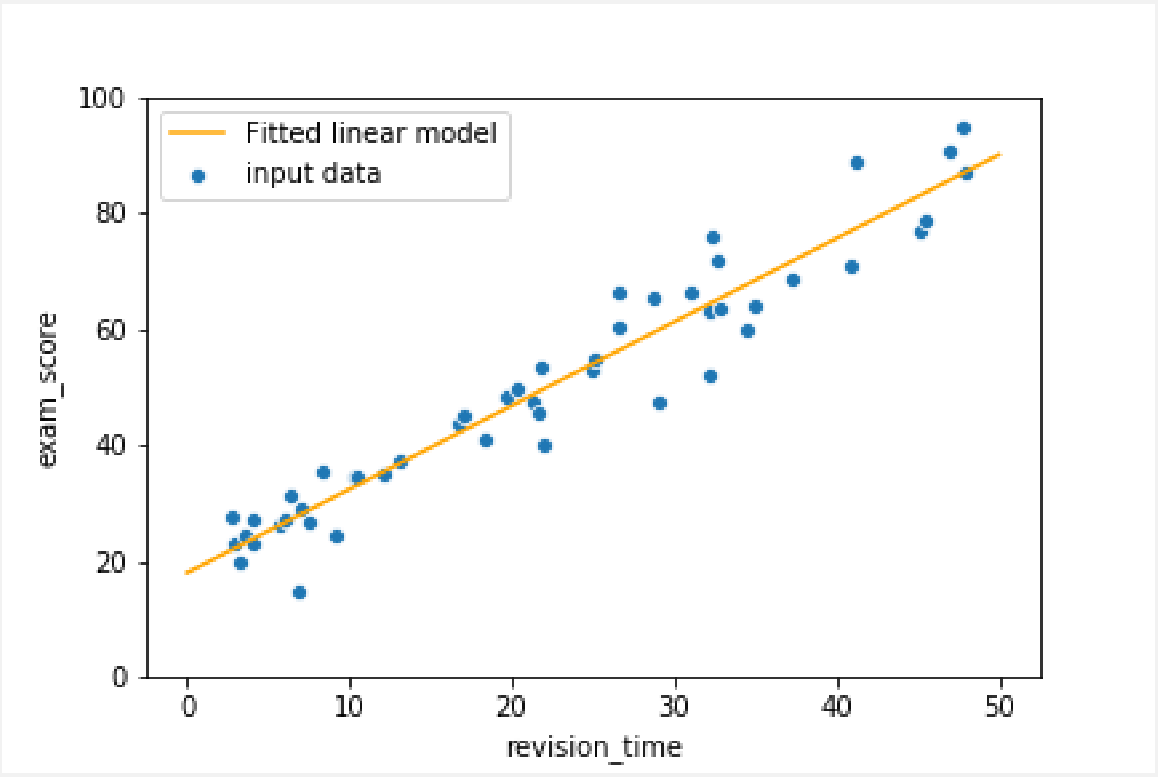
Question 1 continued on next page...

(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 $\mathbf{w}$. 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* $\boldsymbol{\phi}(x_i) = [1, x_i]$.

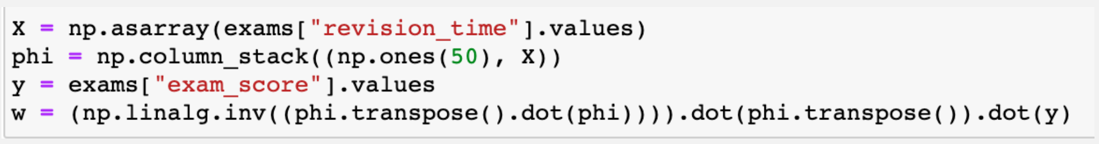
The model estimated $\mathbf{w}_0$ to be 17.89768 and $\mathbf{w}_1$ to be 1.44114. This represents the best fit for the linear model being a line with equation $y = 1.44114 * x + 17.89786$ or, in terms of our attributes, that the best estimate for exam score is 1.44114 times time spent revising, plus 17.89786.

Question 1 continued on next page. . .

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



(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.*



```
X = np.asarray(exams["revision_time"].values)
phi = np.column_stack((np.ones(50), X))
y = exams["exam_score"].values
w = (np.linalg.inv((phi.transpose().dot(phi)))).dot(phi.transpose()).dot(y)
```

This code output the same values for **w** that I reported in question 1b

(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.

$MSE = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$

The main limitation of MSE is its sensitivity to outliers

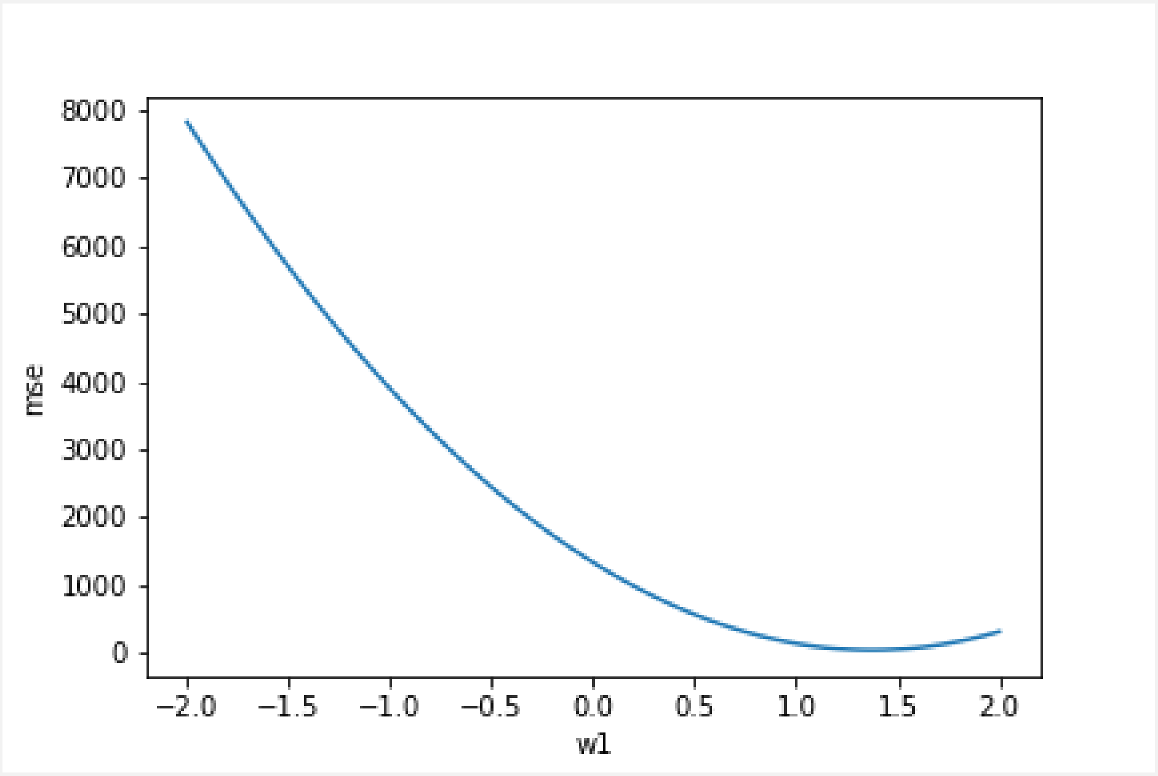
Question 1 continued on next page...

(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 both, to 5 decimal places, was 30.98547. They had identical performance up to 9 decimal places, beyond which skLearn performed better (but this is not really meaningful)

Question 1 continued on next page. . .

(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)`.

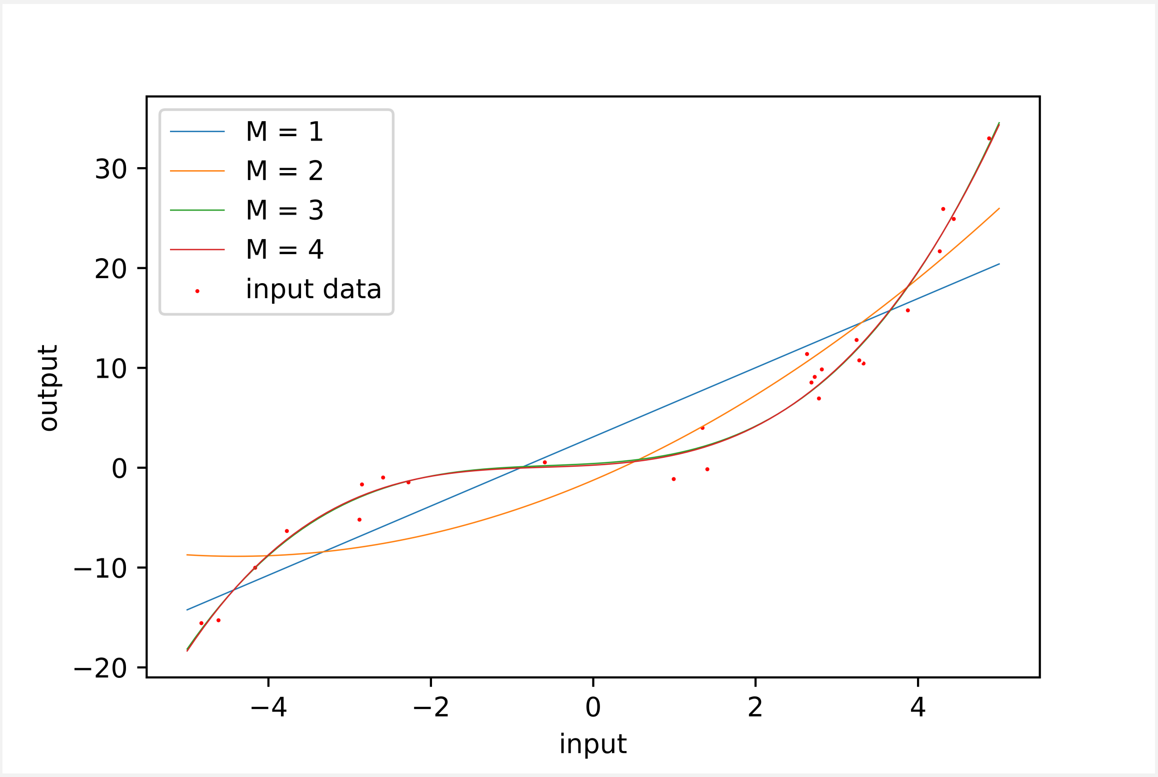


The MSE with respect to w1 formed a parabola with a minimum around 1.35 (The lowest MSE value came, to 5 d.p, at 1.35353). This value is very close to the optimal that we calculated previously, which was to be expected as there was only a small change in w0, and a change **was** to be expected, as using MSE will result in us fitting to the mean, not the pattern.

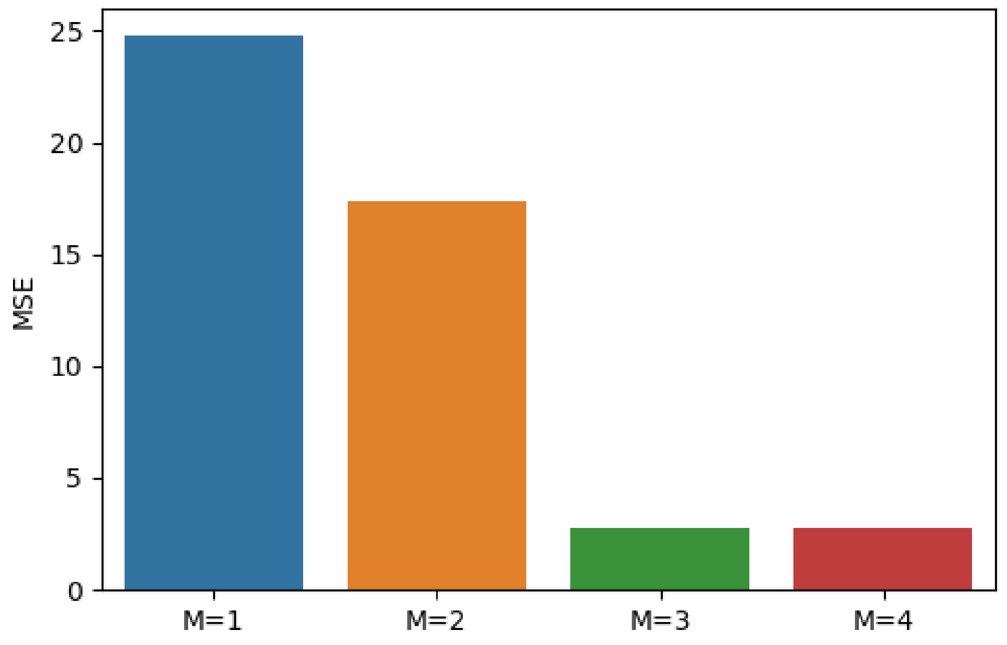
Question 2 : (18 total points) Nonlinear Regression

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`.



(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.

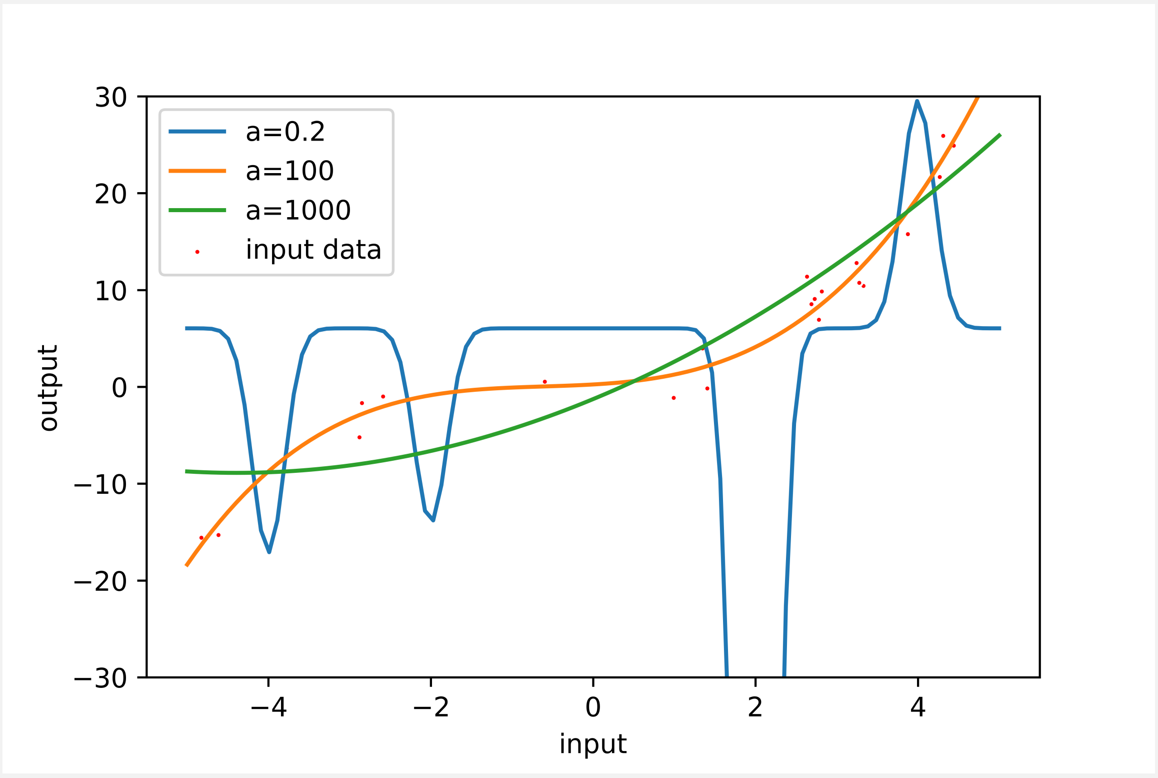


(c) (4 points) Comment on the fit and Mean Squared Error values of the $M = 3$ and $M = 4$ polynomial regression models. Do they result in the same or different performance? Based on these results, which model would you choose?

Both models have achieved the almost identical same performance based on the mean square error. Based on this, I would choose model M=3. Overfitting is not a consideration, as when graphed both models follow almost exactly the same line, however using M=3 over M=4 reduces computational complexity.

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 α. Note that in this example, we are using the same width α 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 α.



(The drop around 2 goes down to around -150)
As alpha increases, the model tends towards a flat, straight line, whereas values of alpha closer to 0 result in a flat line with 'spikes' up or down.

Question 3 : (26 total points) Decision Trees

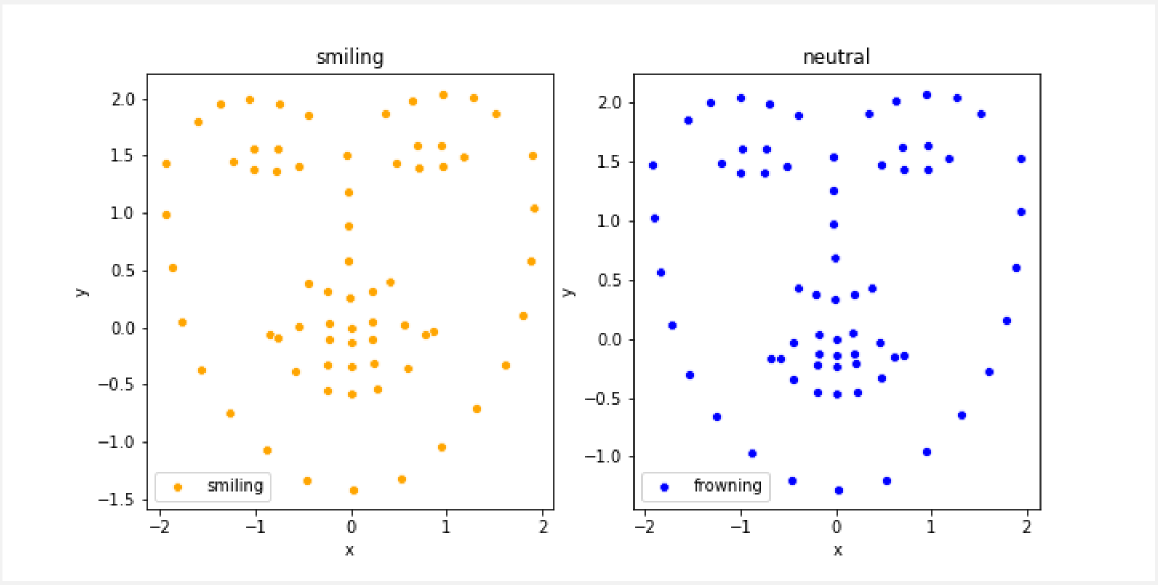
In this question we will train a classifier to predict if a person is smiling or not.

(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 split contains 4800 data points, and the test split contains 1200, with each data point containing 67 x and y values. All individual attributes are continuous. A comparison of mean and std values suggests that attributes of the splits are identically distributed. Distplots of the attributes suggest that they are, in general, normally distributed. The training data is has roughly even numbers of smiling and not smiling instances.

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.



As expected, the face created from rows where smiling=1 appears to be smiling, and the other appears neutral. The main different can be seen at the points that make up the corners of the mouth, which on the smiling plot are further from x=0 and closer towards y=0. Another difference can be seen in the points that make up the bottom of the upper lip and the top of the lower lip, with them being closer and further away from y=0 respectively

Question 3 continued on next page. . .

(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?

The default measure is Gini impurity, which measures the probability that a randomly chosen element from the training set would be incorrectly labeled by the tree (assuming that non-pure leaf nodes assign a random label according to the distribution of classes it contains).
According to the study "Theoretical comparison between the Gini Index and Information Gain criteria" (L. E. Raileanu, K. Stoffel, 2004) there is almost no difference in performance (in terms of ability to classify) between the two methods.

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.

Increasing this parameter will cause the model to fit the training data more tightly, and decreasing will have the opposite effect.
Too small a value could lead to the model being underfit, whereas too large a value would have the opposite effect and cause overfitting. In both instances this would lead to a decrease in accuracy on test and actual input data when compared to a model with an intermediate value

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.

dt_2 (where dt_n is the decision tree generated with max depth set to n) has depth of 2, training accuracy of 0.79479, and test accuracy of 0.78167
dt_8 has depth of 8, training accuracy of 0.93354, and test accuracy of 0.84083
dt_20 has depth of 17, training accuracy of 1.0, and test accuracy of 0.81583

dt_8 is the best model, as it has the highest test accuracy. dt_2 is underfit, as it performs worse on both test and training data, whereas dt_20 is overfit as, while it performs better on training data, it performs worse on test data.

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`.

The most important attributes are x50, y48, and y29.
Plotting the specific attribute in a different color on our plots of mean values demonstrates to us that it corresponds to one of the points that makes up the mouth. This makes sense within the context of trying to detect smiling faces

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.

We cannot make use of the relationships between attributes. Being able to look at the slope of a line between two points would be very useful for detecting smiling (for example, the ends of the mouth on a smiling face are slanted upwards - see the mean plots from earlier)

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?

The accuracy values as percentages are as follows:
alg_1: 38.4, alg_2: 45.0, alg_3: 67.9, alg_4: 67.1

The limitation of this metric is that it is not useful when we have imbalanced class sizes. In our dataset, there are 202 inputs with gt value 1 and 798 inputs with gt value 0 - one could achieve 79.8% accuracy by always predicting 0, but this would clearly not be a useful classifier.

We could two accuracy values for each model, one for each ground truth value. This would make the accuracy values more useful.

Question 4 continued on next page. . .

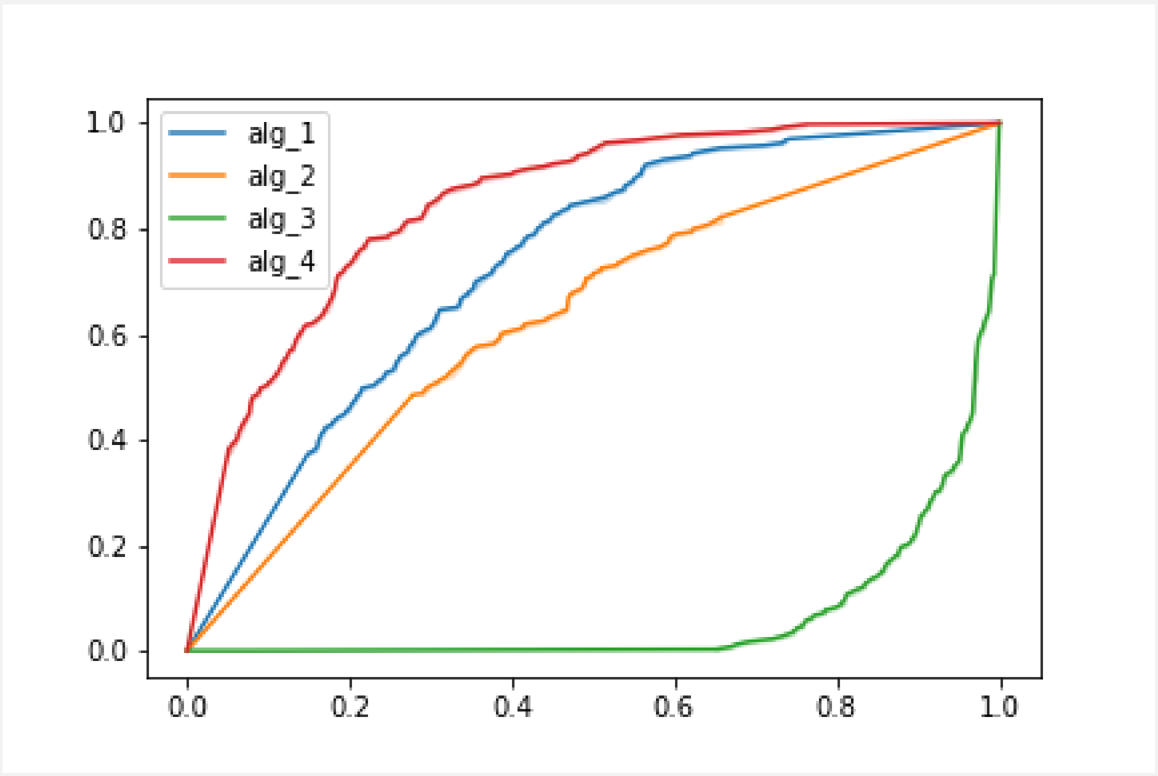
(b) (4 points) Instead of using classification accuracy, report the Area Under the ROC Curve (AUC) for each model. Does the model with the best AUC also have the best accuracy? If not, why not?
Hint: You can use the roc_auc_score function from sklearn.

The AUC scores are as follows:
alg_1: 0.73209, alg_2: 0.63163, alg_3: 0.06395, alg_4: 0.84739

The model with the best accuracy, alg_3, does not have the best AUC score; This is because while it is more accurate for those inputs with a ground truth value of 0, it is less accurate for models with a ground truth value of 1; It has a poor true positive rate, and AUC is directly correlated with this metric

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.



alg_3's performance could be improved by increasing the threshold