IAML – INFR10069 (LEVEL 10): Assignment #1

s1828233

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.

Our data has 2 attributes, `exam_score` and `revision_time` and has 50 samples. All the data is numerical and has the type float. The maximum value for `revision_time` is 48.011 and the minimum is 2.723 giving the data a range of: 45.288. The maximum value for `exam_score` is 94.945 and the minimum is 14.731 giving the data a range of: 80.214.

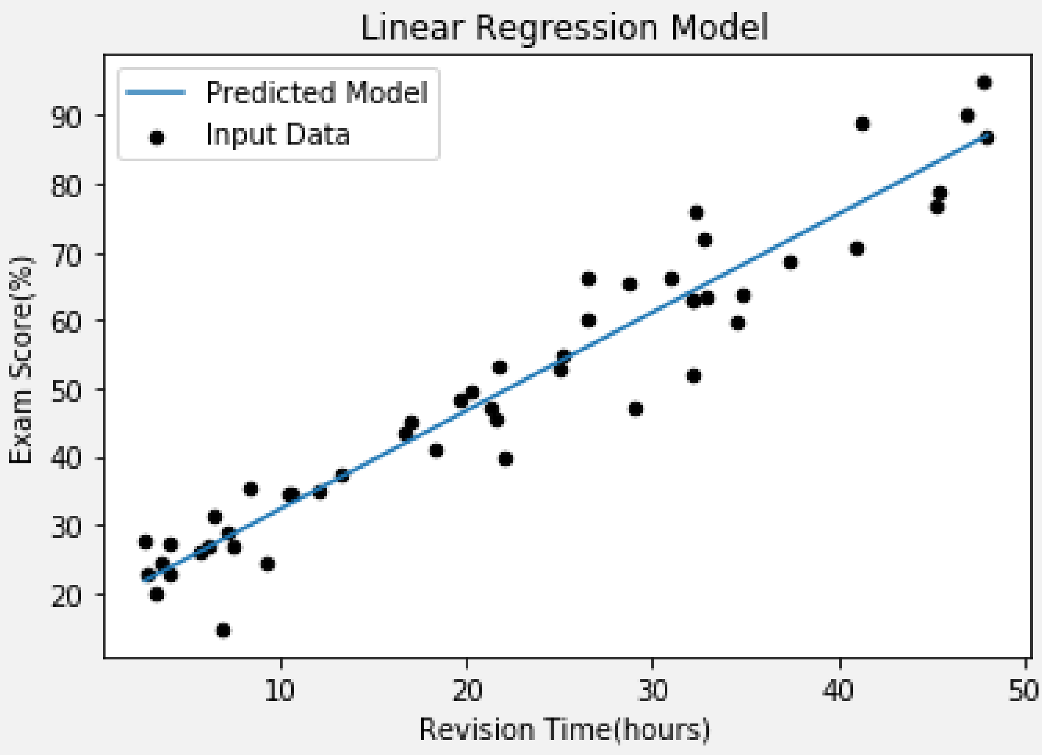
Question 1 continued on next page. . .

(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 estimated parameters are: w = [17.89768026, 1.44114091]. For 1D data w_0 represents the y-intercept of the fitted linear model and w_1 represents the gradient of the fitted linear model. So in this case w_0 is the predicted score for someone who spends 0 time revising, and w_1 represents the predicted rate of improvement in `exam_score` with more `revision_time`.

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 $\mathbf{w}$ *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.*

```
Xt = np.transpose(X)
weights = (np.dot(np.linalg.inv(np.dot(Xt, X)), np.dot(Xt, y)))
print(weights)
```

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

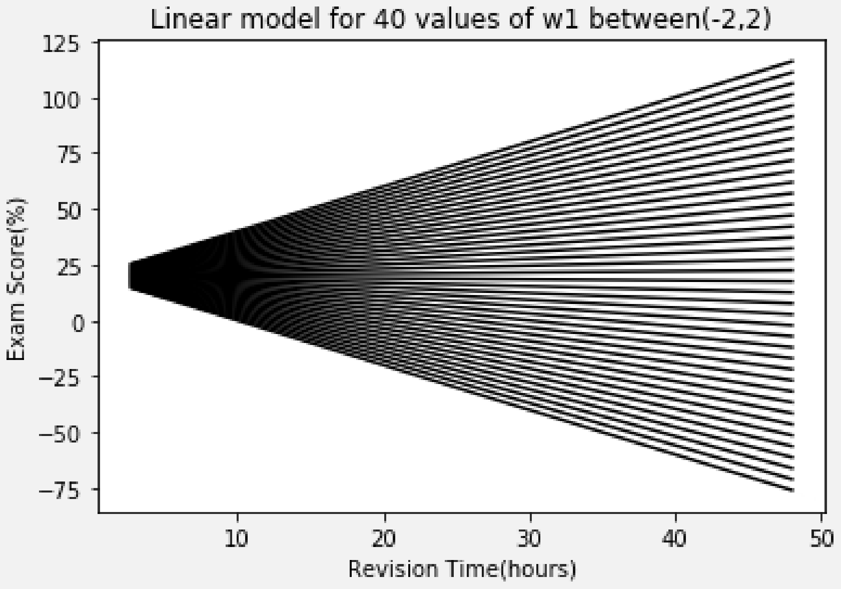
A limitation of MSE is that it is prone to outliers as it uses the concept of using mean in computation of each error value.

(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 Mean Squared Error for both the sklearn model and the closed-form solution model was identical, both returning a value of 30.99(3.s.f). Since the Mean Squared Error was the same for both models, it is safe to say they both performed equally well.

(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)`.

w_1	MSE
-2.000	7830.38
-1.897	7363.34
-1.795	6910.73
-1.692	6472.53
-1.590	6048.76
-1.487	5639.41
-1.385	5244.48
-1.282	4863.97
-1.179	4497.88
-1.077	4146.21
-0.974	3808.96
-0.872	3486.13
-0.769	3177.72
-0.667	2883.74
-0.564	2604.17
-0.462	2339.02
-0.359	2088.30
-0.256	1851.99
-0.154	1630.11
-0.051	1422.65
0.051	1229.60
0.154	1050.98
0.256	886.78
0.359	737.00
0.462	601.64
0.564	480.69
0.667	374.18
0.769	282.08
0.872	204.40
0.974	141.14
1.077	92.30
1.179	57.89
1.282	37.89
1.385	32.31
1.487	41.16
1.590	64.42
1.692	102.11
1.795	154.22
1.897	220.75
2.000	301.69

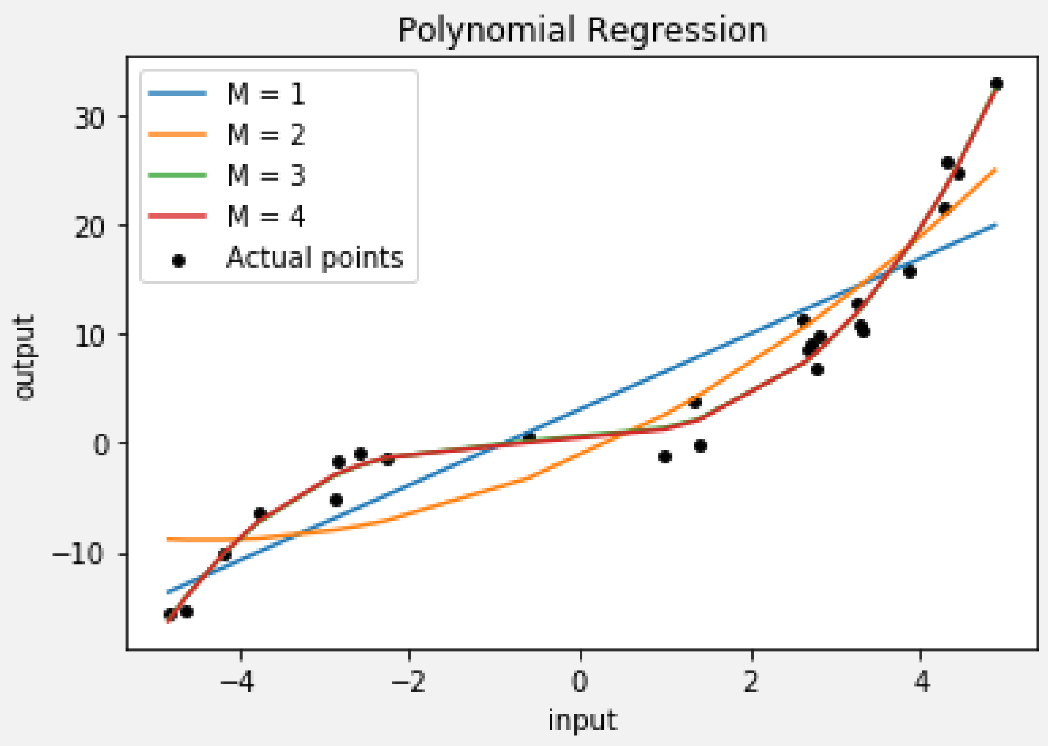


The plot looks like a projection of light from a fixed point, which in this case was w_0. The MSE is at it's minimum when $w_1 \approx 1.38$. This value is expected since we expect a positive relationship between `revision_time` and `exam_score`.

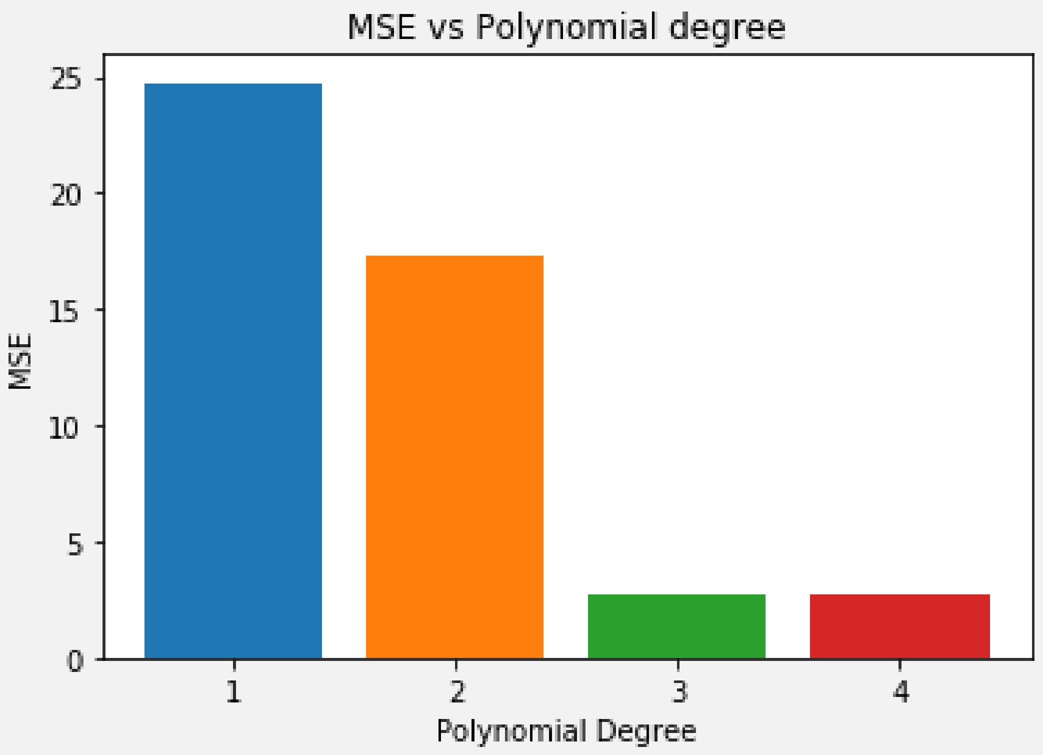
Question 2 : (18 total points) Nonlinear Regression

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.

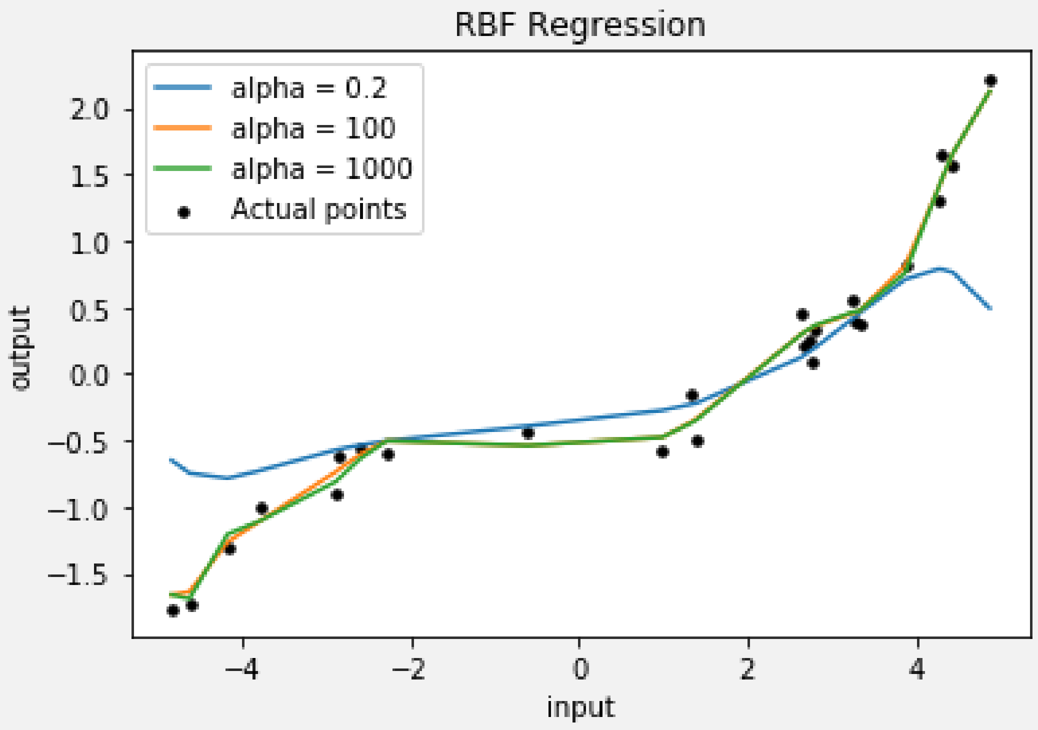


(c) (4 points) Comment on the fit and Mean Squared Error values of the $M = 3$ and $M = 4$ polynomial regression models. Do they result in the same or different performance? Based on these results, which model would you choose?

The fit and Mean Squared Error is the same for M = 3 and M = 4. Since the performance is the same for both models, I would chose the model with M = 3, since it is the simpler model, meaning it will be faster to compute. If trained on another set of data the model when $M = 3$ is also less likely to overfit that data than the model when $M = 4$.

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 value of α appears to affect how closely the regression model fits the data and the range of the highest and lowest values predicted. A low alpha results in the range of values predicted to be much lower and the model is less fitted to the data. Higher values result in a better fit and wider range of values predicted.

Question 3 : (26 total points) Decision Trees

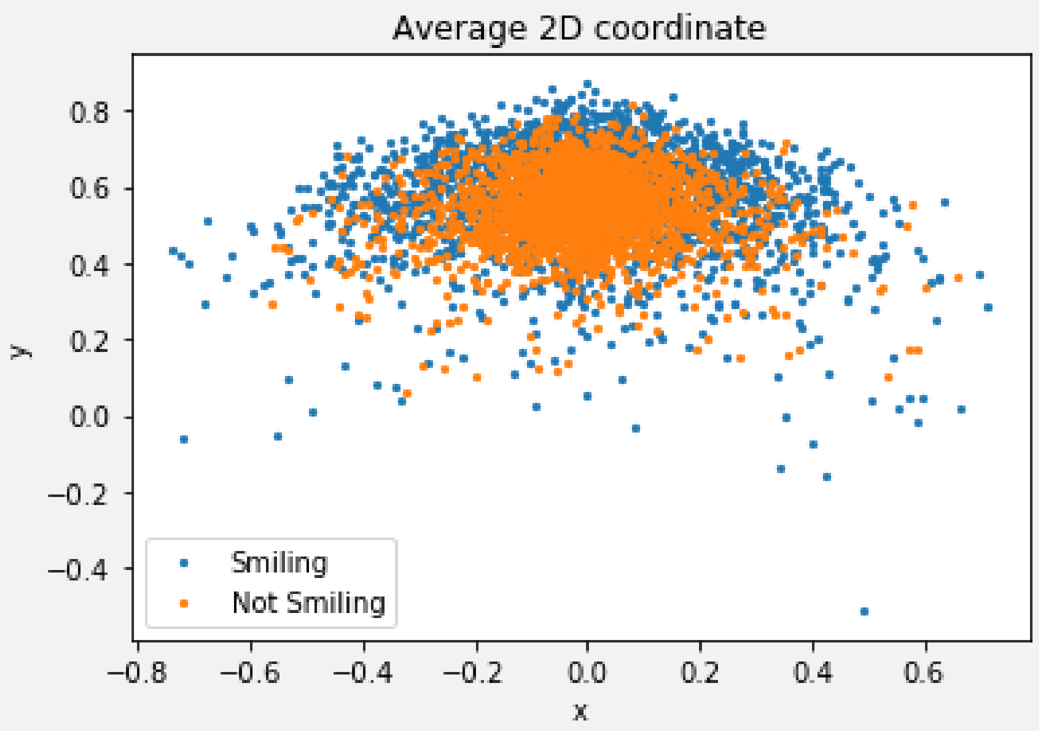
In this question we will train a classifier to predict if a person is smiling or not.

(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 original test and training data have 137 attributes each, but one is the target class label of type int, so it has been separated from the training and test splits. This means that the training split has 4800 samples and 136 attributes, while the test split has 1200 samples and 136 attributes. All the data in the test and training splits is numerical and is of the type float.

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.



Based on the plot above,the average coordinates for the smiling faces appear to be more stretched out, along both the x and y axes.

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?

DecisionTreeClassifier uses Gini impurity as the default measure of purity at each node. The main advantage of Gini impurity over entropy is that it doesn't involve using logarithms in the computation of the node purity, while entropy does. This means that using the Gini impurity will be faster and less computationally expensive.

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.

Depth of a decision tree effects how well our model fits the training data, with a higher depth the model fits the data better and vice-versa. One problem of low depth decision tree classifier is that it will most likely result in underfitting to the data. While an issue of a high depth is that it can cause overfitting. Another drawback of a high depth is that decision tree training is very slow and expensive, so a a large max depth will only exaggerate this problem.

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.*

Maximum Depth	Train Accuracy	Test Accuracy
2	79.5%	78.2%
8	93.4%	84.1%
20	100%	81.6%

Based on these results, the best model is the one with a maximum tree depth of 8. The reason it's the best is because it provides the best balance of not underfitting or overfitting to the data and it is faster to compute than the model with a maximum tree depth of 20.

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 is x_{50} with a value of 0.330, the second most important is y_{48} with a value of 0.0900 and the third is y_{29} with a value of 0.0883. The highest attribute doesn't really make sense in this classification task because we need a whole attribute vector consisting of x,y coordinate pairs to identify if a face is smiling or not so, one individual x coordinate does not help us define whether a face is smiling or not, as without the y coordinate it is not even enough to pinpoint a position in the 2D space.

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 limitations to using 2D input attributes in this classification. Since a persons face is 3D, using only 2D to encode points on a persons face means we lose information, in this case depth. This can affect classification results, since a person smiling also means the depth of certain points on the face change, but in 2D we don't have this information, so classification is based purely on x,y coordinates.

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?

Algorithm	Classification Accuracy
1	61.6%
2	55.0%
3	32.1%
4	32.9%

Using the current metric, the best model is the model of algorithm 1. Using a threshold default threshold (>=0.5) can result in poor performance if the dataset has a class imbalance. So one improvement to this would be to adjust the threshold to suit the data being used, which will help account for some class imbalance.

Question 4 continued on next page...

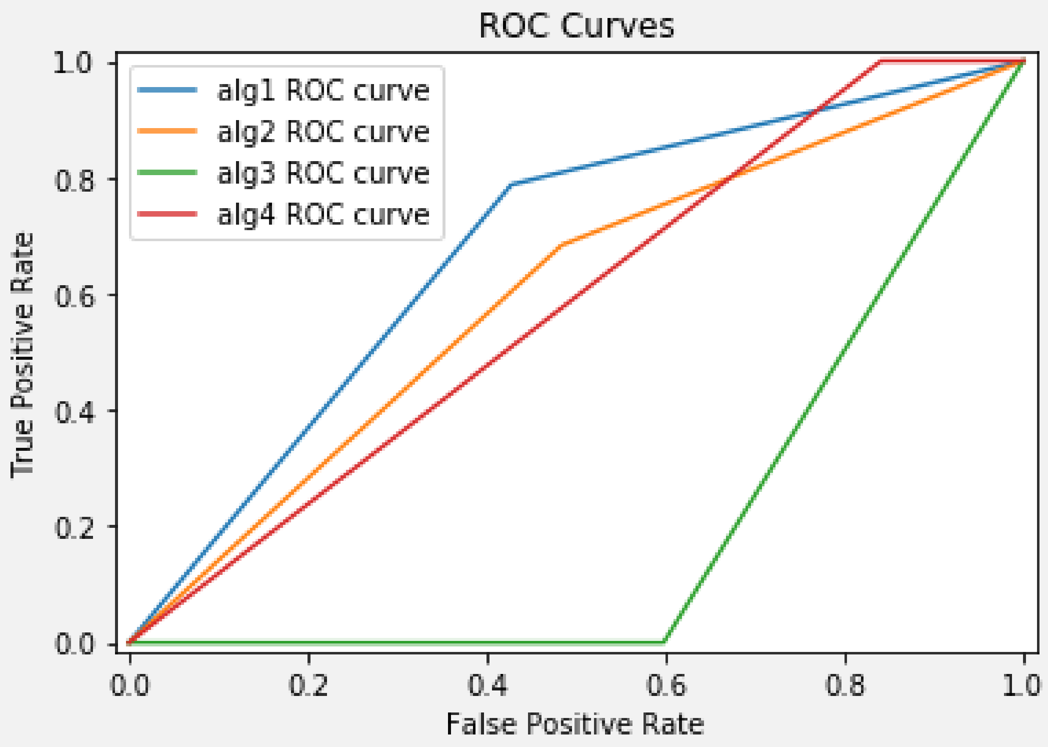
(b) (4 points) Instead of using classification accuracy, report the Area Under the ROC Curve (AUC) for each model. Does the model with the best AUC also have the best accuracy? If not, why not?
Hint: You can use the roc_auc_score function from sklearn.

Algorithm	Area Under ROC Curve
1	0.732
2	0.632
3	0.064
4	0.847

The model with the best AUC does not have the best accuracy. This is because AUC and accuracy are 2 different measures. Accuracy is calculated based on predicted class vs true classes with a single threshold, while AUC is an evaluation of the classifier as threshold varies over all possible values. This means AUC can be considered a wider testing 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.



The ROC curve for algorithm 3 is completely different from the other algorithms. It is completely flat until the false positive rate reaches a value of 0.6, after which the curve shoots up. Without retraining it will be very difficult to improve the performance of algorithm 3.