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

Alexandru-Ioan Chelba

# 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 dataframe has 50 rows of data given in 2 columns: revision_time and exam_score. The data type used is float64 in both columns. The data in revision_time ranges from 2.723 to 48.011, while in exam_score it ranges from 14.731 to 94.945.

*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 model parameters are:

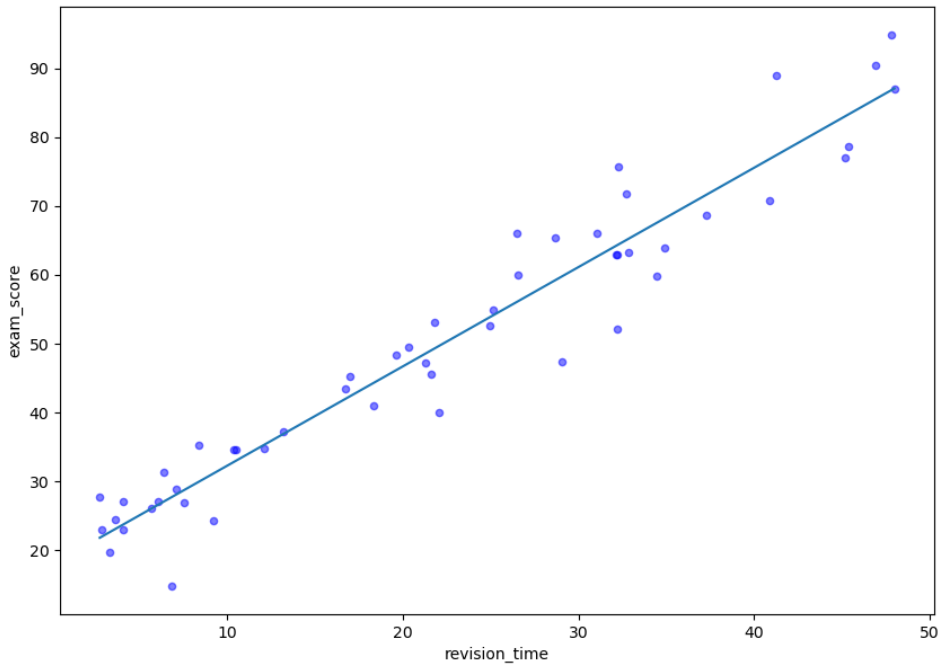
$$\mathbf{w} = \begin{bmatrix} 17.89768026 \\ 1.44114091 \end{bmatrix}$$

The first parameter of $\mathbf{w}$ is the bias, while the second one is the gradient.

*Question 1 continued on next page...*

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

This image shows the fitted linear model as one line, and the input data as dots:




*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* $\mathbf{w}$ *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.*

```
# Calculate the mean of X and y
xmean = np.mean(X)
ymean = np.mean(y)

# Calculate the terms needed for the
# numator and denominator of beta
xycov = (X - xmean) * (y - ymean)
xvar = (X - xmean)**2

# Calculate the gradient and the bias
gradient = xycov.sum() / xvar.sum()
bias = ymean - (beta * xmean)
y_pred = bias + gradient * X
```

*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 ŷ (`$\hat{y}$` in latex) in place of the model prediction.*

The formula for Mean Squared Error is:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(Y_i - \hat{Y}_i)^2$$

It has the limitation of heavily weighting outliers. This is a result of the squaring of each term, which effectively weights large errors more heavily than small ones.

*Question 1 continued on next page...*

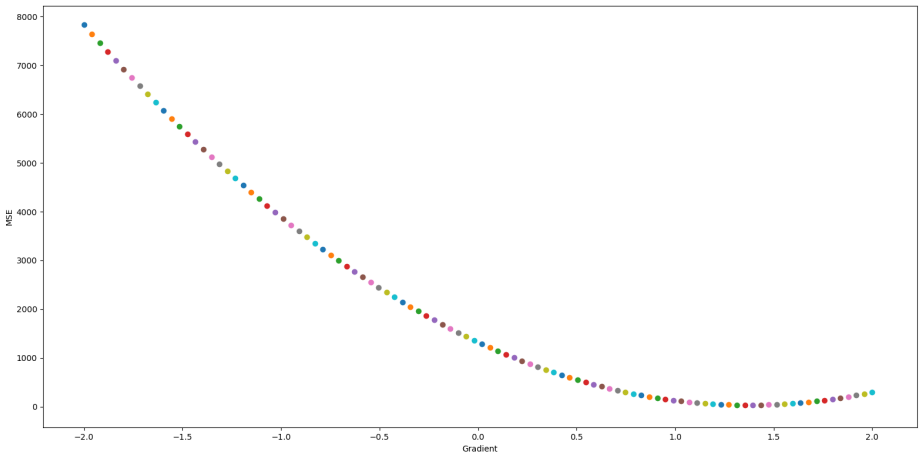
(f) (3 points) Our next step will be to evaluate the performance of the fitted models using Mean Squared Error (MSE). Report the MSE of the data in `regression_part1.csv` for your prediction of `exam_score`. You should report the MSE for the linear model fitted using sklearn and the model resulting from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

The MSE of the data for my prediction of exam_score is 30.9854726145413, while the MSE for the model resulting from my closed-form solution is 30.985472614541287. The difference is almost non-existent and probably due to approximation errors.

*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 required plot is:



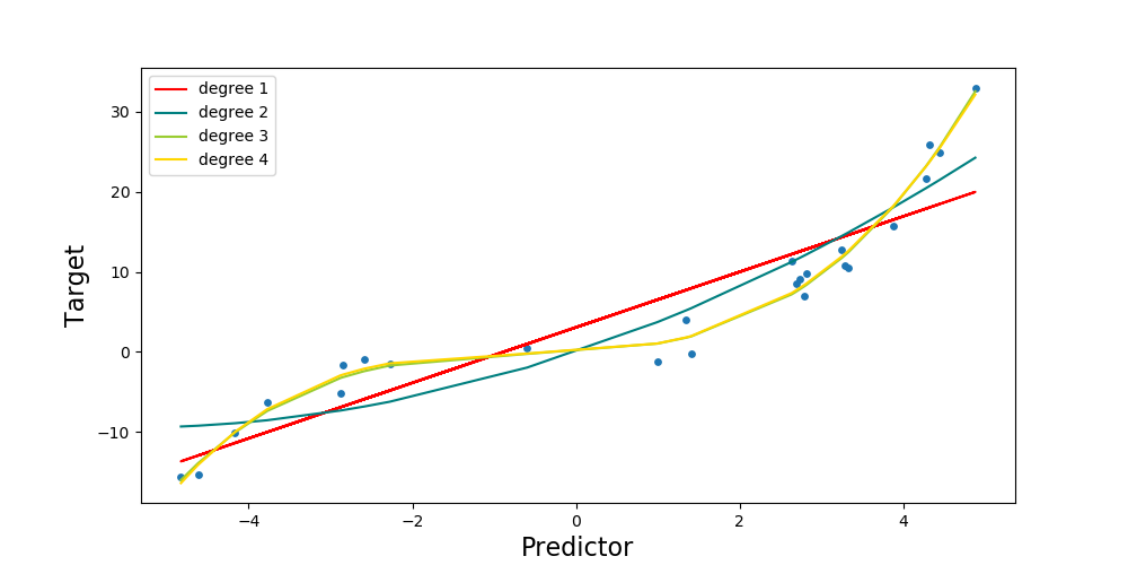

Its minimum is at 1.(35). It is close to the value w1 has in sub-point b of this exercise, which is to be expected. In addition, I expect it to be smaller than that value, since I increased the bias.

# 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`.

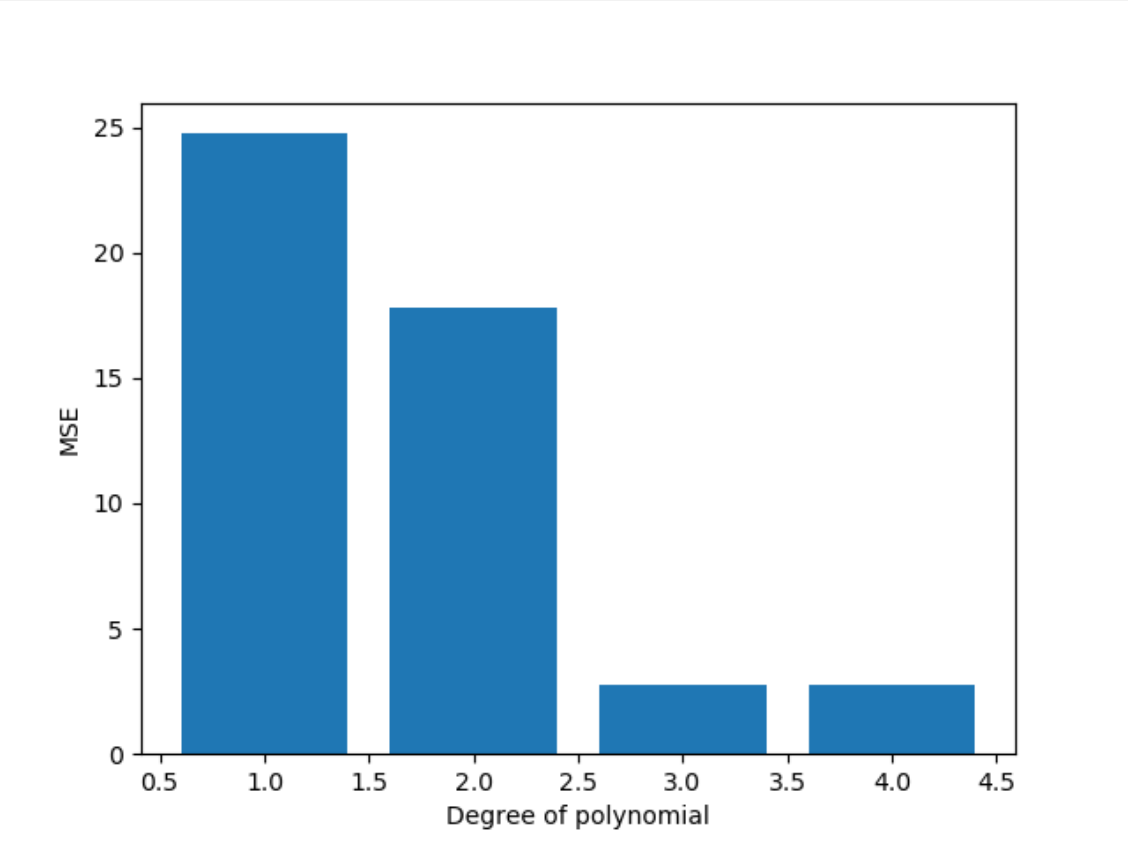
The plot is:




*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.

The plot is:




The values are: 24.744378032672856, 17.796426891430183, 2.7879149846359774, 2.7460066969556256.

*Question 2 continued on next page...*

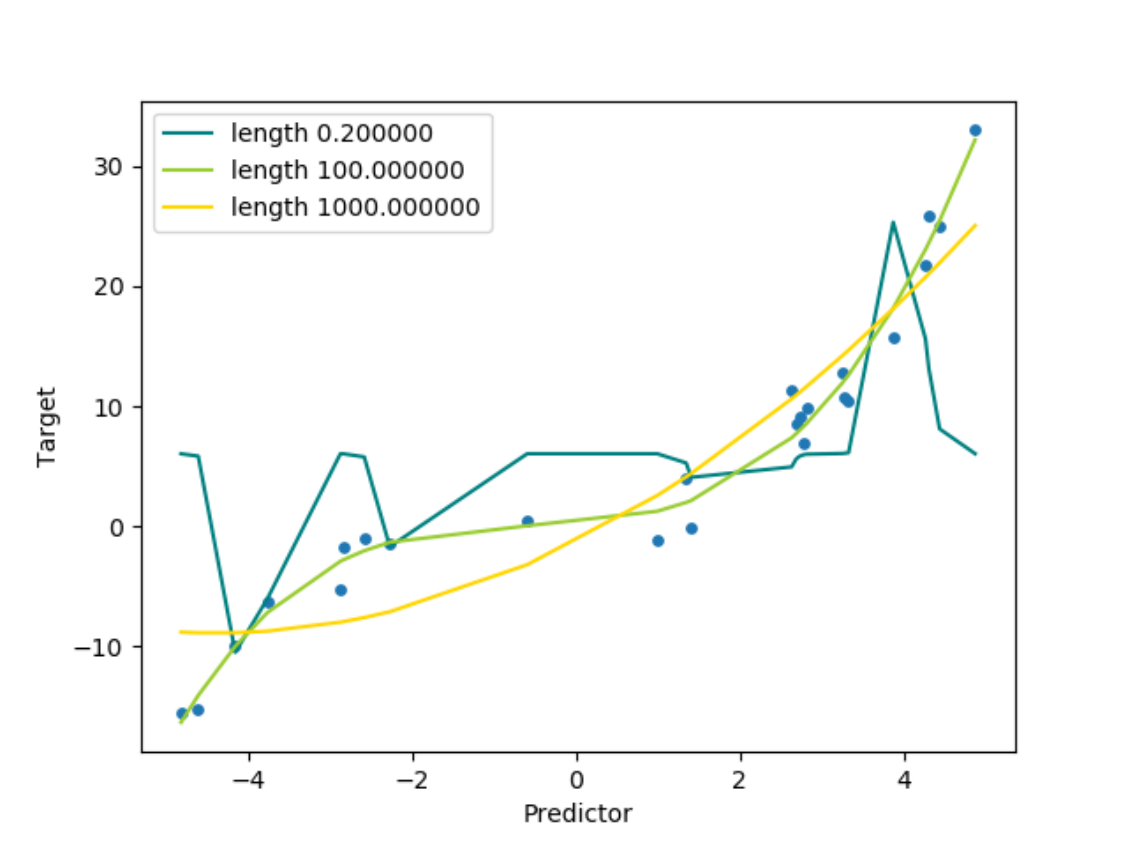
(c) (4 points) Comment on the fit and Mean Squared Error values of the $M = 3$ and $M = 4$ polynomial regression models. Do they result in the same or different performance? Based on these results, which model would you choose?

They result in approximately similar performances: for $M = 3, MSE \approx 2.78$ and for $M = 4, MSE \approx 2.74$. I would still choose the model for $M = 3$, because the model for $M = 4$ is better only by a tiny margin, which suggests that it overfits the training data.

*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 $\alpha$.

The plot is:




A bigger value of $\alpha$ enforces the component to be smoother.

# Question 3 : (26 total points) Decision Trees

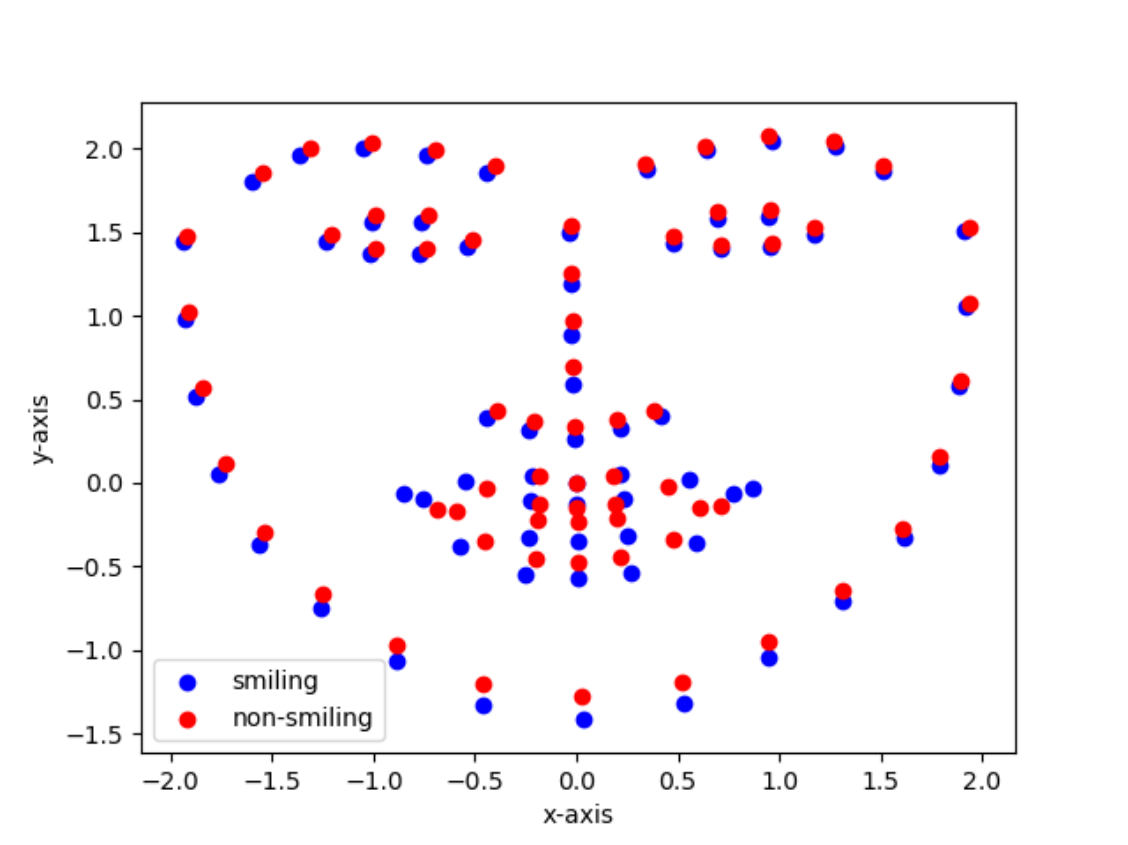
**In this question we will train a classifier to predict if a person is smiling or not.**

(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.

There are 4800 rows and 136 columns (excluding 'smiling') of float64 data in the training set, while in the test set there are 1200 rows and 136 columns (excluding 'smiling') of float64 data.

*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 points that describe the lips have a different average positioning for smiling and non-smiling faces. Another notable difference can be seen for the points describing the chin and the nose, where smiling faces have the points a bit lower-positioned than the non-smiling faces.

*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?

Advantages of Gini are that it is less computationally intensive than entropy (no logarithm to be calculated).

*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.

If the maximum depth is too small, then the model will underfit my training data, because it will not be capable to find all the defining parameters. On the other hand, if the maximum depth is too high, then the model will overfit my training data, because it will learn the specifics of the data as well.

*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.*

The following table describes the training accuracy and test accuracy for maximum depths of 2, 8 and 20:

| Maximum depth | 2 | 8 | 20 |
|---|---|---|---|
| Training Accuracy | 79.48% | 93.35% | 100.0% |
| Test Accuracy | 78.17% | 84.08% | 81.58% |

The best model is the one with maximum depth of 8, since it has good accuracy on the training set as well as the best accuracy for the test set. Although the tree with maximum depth of 20 has 100% accuracy on the training accuracy, it has a worse accuracy on the test set than the tree with maximum depth of 8, which suggests it overfits the training 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 three most important attributes, according to the Gini importance, are: $x_{50}$, $y_{48}$, $y_{29}$. The one with highest importance makes sense, since $x_{50}$ represents the x-coordinate of one upper-lip point, and it differs significantly from a smiling face to a non-smiling face.

*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.

One limitation of 2-dimensional view is that, if the face is plotted from a non-straight angle, there is no way to actually rotate the face or change the perspective because the image lacks depth.

# 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?

For a threshold of >=0.5, we have the following accuracies:

| Algorithm | alg_1 | alg_2 | alg_3 | alg_4 |
|---|---|---|---|---|
| Accuracy | 61.6% | 55.0% | 32.1% | 32.9% |

According to this metric, the best model is alg_1. A limitation is that accuracy does not take into account whether the classes are balanced or not. Ways to improve accuracy include making sure the training set is unbiased, or using different weights for positives and negatives.

*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 area under the curve for each algorithm can be seen below:

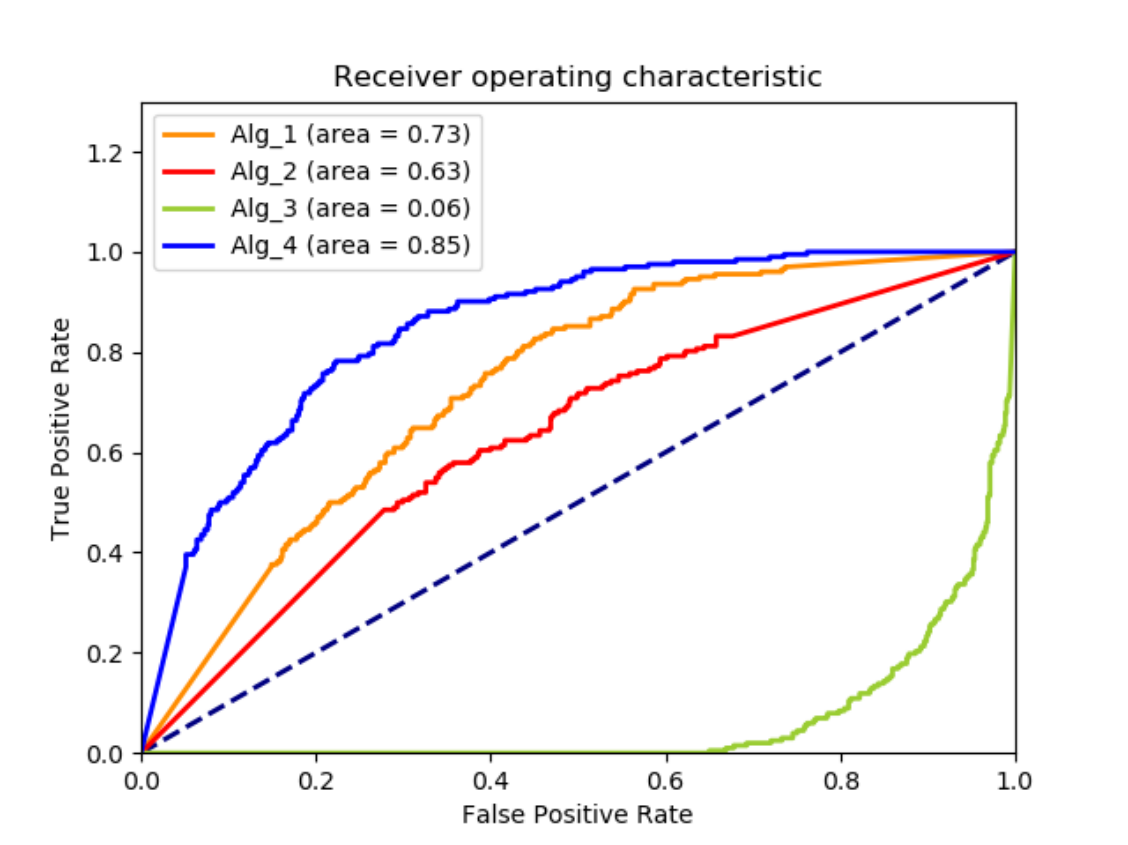
| Algorithm | alg_1 | alg_2 | alg_3 | alg_4 |
|---|---|---|---|---|
| AUC | 0.73 | 0.63 | 0.06 | 0.85 |

The model with the best AUC is alg_4, while the model with the best accuracy is alg_1. The reason they are different is that AUC takes into account all possible threshold values, while accuracy only take into account a threshold of >=0.5.

*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 plot is:




One way to improve the performance of alg_3 without having to retrain the model is to swap the number of true positives with the number of false positives. This will result in a mirrored line of the green line on the dashed line.