IAML – INFR10069 (LEVEL 10): Assignment #1

s1703084

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.

In the dataset of regression_part1.csv, there are 50 observations (rows) and 2 attributes (columns). The columns of 'revision_time' and 'exam_score' are both in float64 type. 'revision_time' is in the range from 2.723 to 48.011, and 'exam_score' is in the range from 14.731 to 94.945.

Question 1 continued on next page...

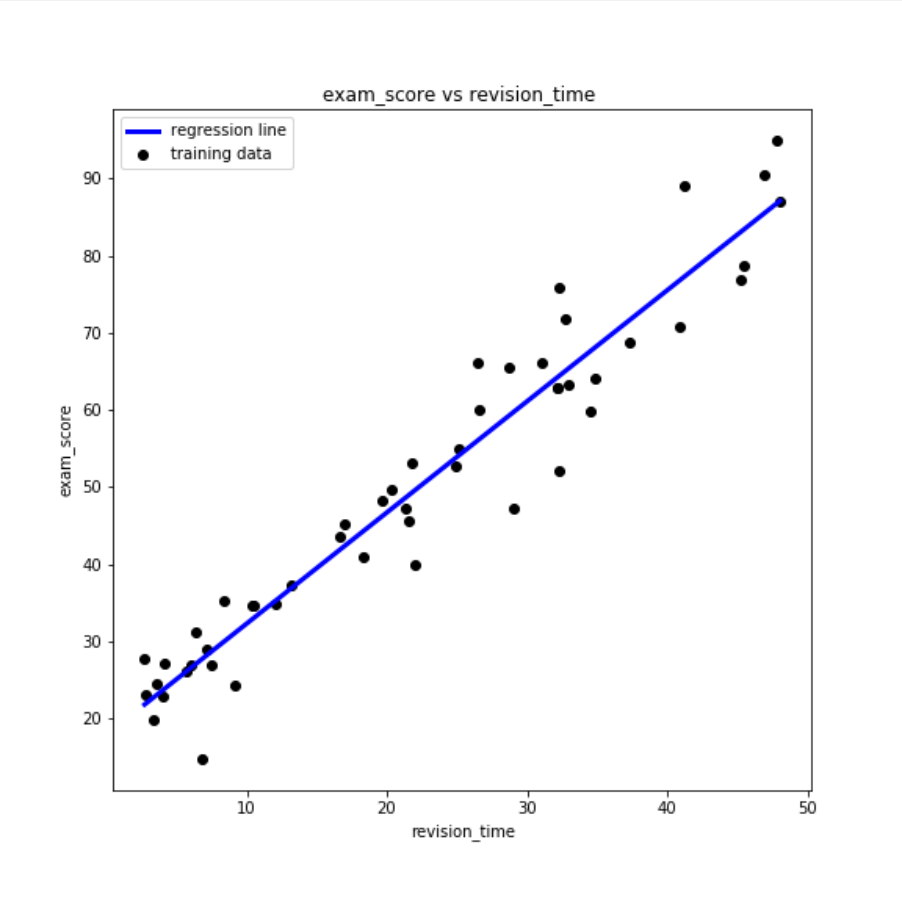
(b) (3 points) Fit a linear model to the data so that we can predict `exam_score` from `revision_time`. Report the estimated model parameters $\mathbf{w}$. Describe what the parameters represent for this 1D data. For this part, you should use the sklearn implementation of Linear Regression.
Hint: By 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 w are [17.89768026, 1.44114091]. w indicates intercept and regression coefficient between a predictor variable and the response. It means when revison_time is 0 (X=0), the expected value of exam_score is around 17.898 (y=17.898); when revison_time increases by an hour, the expected value of exam_score should be increased by around 1.441.

Question 1 continued on next page...

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

The figure below shows the fitted linear model and the input data



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.*

```
from numpy.linalg import inv
coeffs = inv(X.transpose().dot(X)).dot(X.transpose().dot(y))
coeffs
```

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.

$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$
One of limitations of using MSE is very similar as variance, greater errors weights more than smaller errors, giving less tolerance of outliers when evaluating the performance of a model.

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 for the linear model fitted using sklearn is 30.985472614541294.

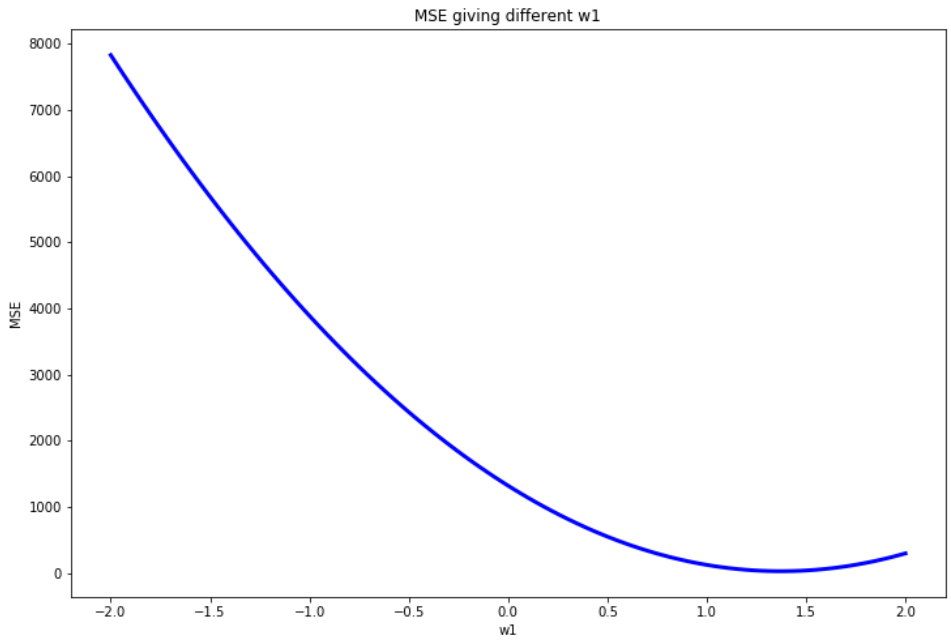
The MSE for this model resulting from the closed-from solution is 30.98547261454129.

They are almost same with tiny difference in approximately 3.553×10^{-15}

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 figure below indicates the relationship between Mean Squared Error (MSE) on vertical axis and different values of w_1 in range from -2 to 2, giving $w_0 = 20$. Following with w_1 increases, MSE decreases until it reaches the minimum point where w_1 is around 1.3. I think it is a expected value, assuming the the vertical shift of regression line is 20 (intercept), we got the slope equal to around 1.3 with minimum ESM. It is very similar as we got from above questions w = [17.89768026, 1.44114091].

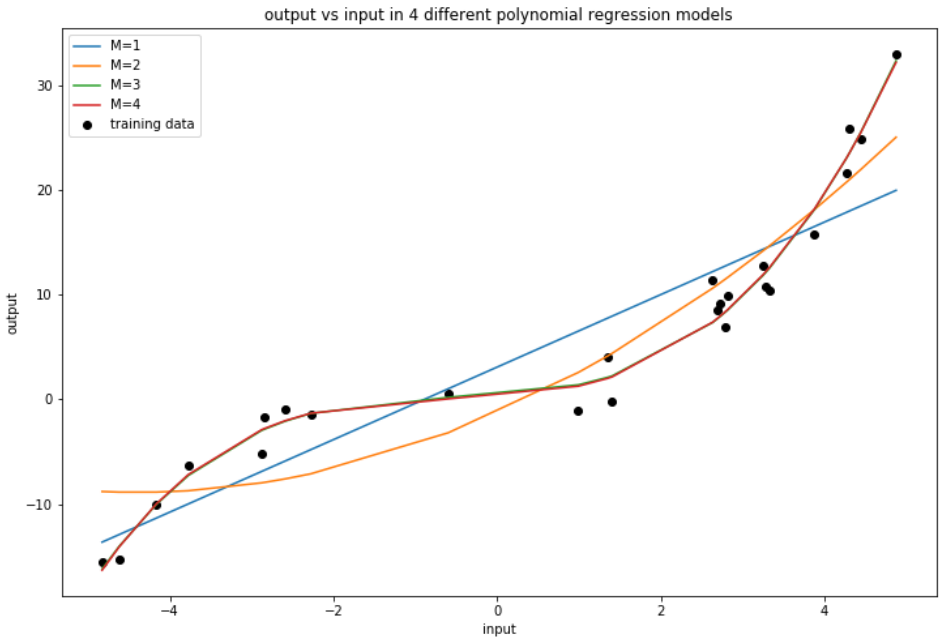


Question 2 : (18 total points) Nonlinear Regression

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`.

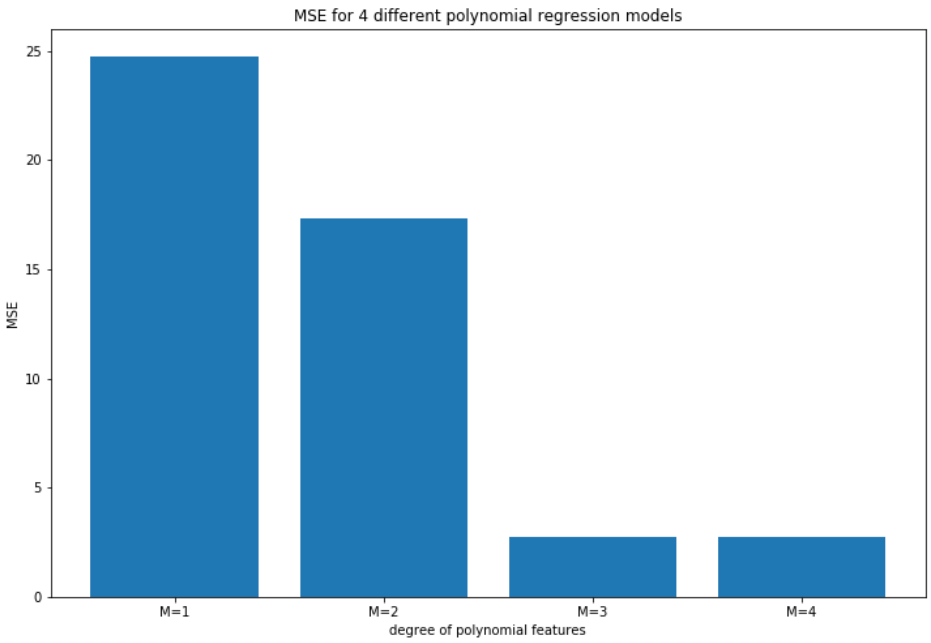
The figure below shows 4 different polynomial regression models by varying the degree of polynomial features from 1 to 4.



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 figure below shows EMS for 4 different polynomial regression models by varying the degree of polynomial features from 1 to 4.



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?

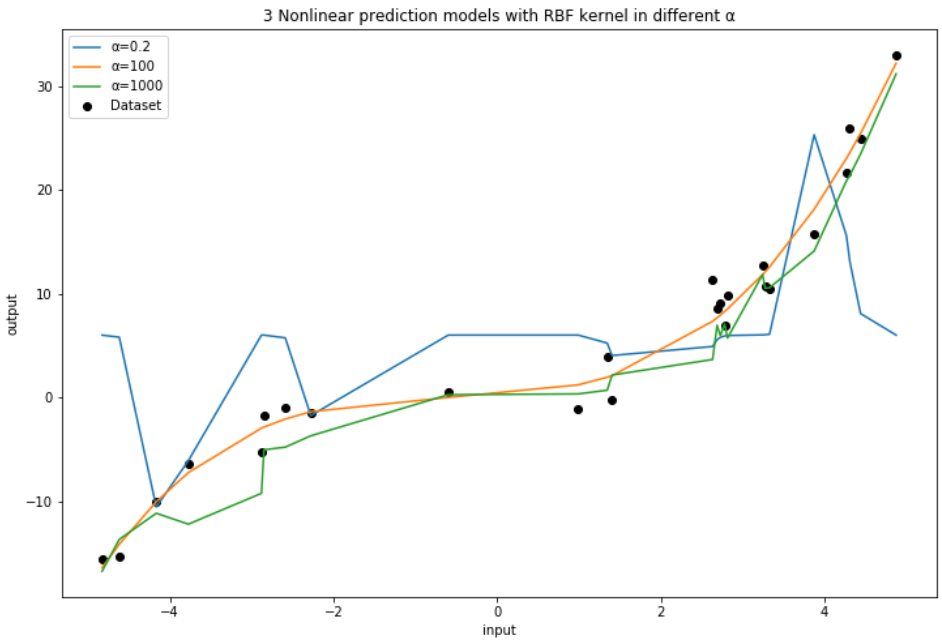
when the polynomial regression model with $M = 3$, the model coefficient is [[0.4053055 0.48562415 0.31115341 0.19142221]], and the ESM is 2.7447567192524263. when the polynomial regression model with $M = 4$, the model coefficient is [[0.24292578 0.48100068 0.35212378 0.19152571 -0.00169485]], and the ESM is 2.7389111790755383.

The two models are almost resulting same performance, and their fit and MSE are very close with tiny difference. I wound choose $M = 3$ polynomial regression model since this model with lower 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 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 figure below shows the nonlinear regression models by using RBF kernel in different α. From the plot, we can see when α is small (for example $\alpha = 0.2$), it can not capture the pattern of the training data and it is underfitting. when α is large (for example $\alpha = 1000$), it is overfitting, and the curve is less smooth than $\alpha = 1000$.



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.

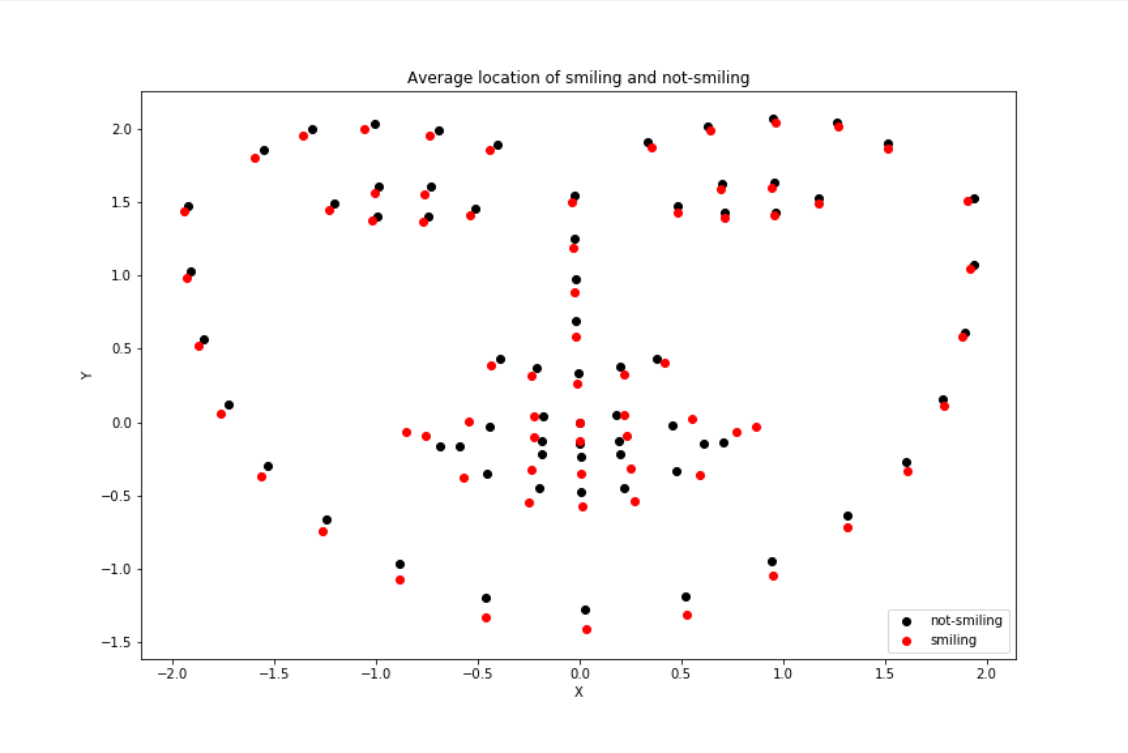
In the training dataset: there are 4800 observations (rows) and 68 pairs of 2D coordinates. These pairs of 2D coordinates are in the type of float64. The attribute of smiling is in the type of int64.

In the testing dataset: there are 1200 observations (rows) and 68 pairs of 2D coordinates. These pairs of 2D coordinates are in type of float64. The attribute of smiling is in the type of int64.

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 figure below represents the average location of smiling face and not smiling face from the training dataset. From the plot we can see, the coordinates of two faces near eyebrows, cheek, eyes and nose are very close and similar. However, the dots near mouth and chin are different in two faces: smiling face have more spread dots than not smiling face.



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?

Gini is the default criteria to measure the quality of a split.

Gini is easier to facilitate the larger distribution than entropy. And Gini is calculated with less computation than entropy

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.

maximum depth of the decision tree is a measure of how many splits a tree can achieve before coming to a prediction.

The lower depth will cause the classifier hard to find the best value (trend to be underfitting). Larger value will increase the complexity and computation of the classifier and may fit perfectly for the training data but not working well on testing data (trend to be overfitting)

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.*

(1)when maximum depth=2, training accuracy=79.479% and test accuracy is 78.167%. (2) when maximum depth=8, training accuracy=93.345% and test accuracy is 84.083%. (3) when maximum depth=20, training accuracy=100% and test accuracy is 81.583%.

I think the model with maximum depth=8 is best. Increasing maximum depth from 2 to 8, the training accuracy and testing accuracy is increasing as well, it means the performance of the model becomes better. But increasing maximum depth from 8 to 20, the training accuracy is increasing while testing accuracy is decreasing, it means the model suffers from overfitting for training data when have larger maximum depth. So, the model with maximum depth=8 have best performance among these 3 models.

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 top three most important attributes are 'x50', 'y48' and 'y29'. I think it make senses. The values in the attribute of 'x50' can be classified as smiling face is from -0.332 to -0.089. And the values in the attribute of 'x50' can be classified as not smiling face is from -0.311 to 0.110. The mean values of two classes is not that close, the overlapping region is not too much.

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.

The coordinates instead of single datapoint should be used to train the model. using the 2D coordinates in pair as input may result in more accurate prediction

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 for alg_1 is 61.6%, for alg_2 is 55%, for alg_3 is 32.1% and for alg_4 is 32.9%. Among these four models, alg_1 is the best since it with the highest accuracy.

Setting the thresholds value to 0.5 may be not resulting in the best performance, and it is meaningless if classes imbalanced. Using the ROC curve and varying the threshold values to look for better performance .

Question 4 continued on next page...

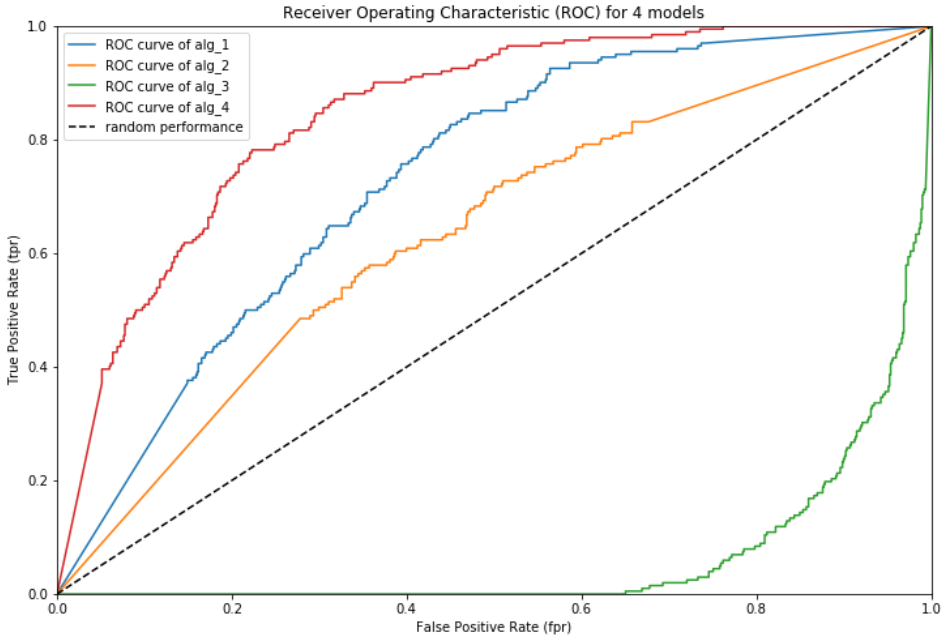
(b) (4 points) Instead of using classification accuracy, report the Area Under the ROC Curve (AUC) for each model. Does the model with the best AUC also have the best accuracy? If not, why not?
Hint: You can use the roc_auc_score function from sklearn.

The AUC for alg_1 is 0.732, for alg_2 is 0.632, for alg_3 is 0.064 and for alg_4 is 0.847.

NO, alg_4 with the highest AUC among these four models but with the quite low accuracy. The accuracy is only focusing the correct predictions of a model and it is meaningless if the classes imbalanced. However, AUC is measuing the overall performance of a model without considering which threshold value is used.

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 of alg_3 is quite strange and looks like opposite to other curves. This ROC curve is more close to lower right, but other curves are more close to top left which is close to perfect performance. Using 1 minus the each element under '`alg_3`' before using roc_curve to plot the curve could improve the performance of `alg_3`.