IAML – INFR10069 (LEVEL 10): Assignment #1

s2001696

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.

According to the given data, there are 50 numbers of data. Both the input and output dimensions are equal to one. Since we concatenate a constant feature 1 to each input feature, the input data serves to $\mathcal{R}^{50*2}$ and the output data for prediction serves to $\mathcal{R}^{50*1}$. The input feature `revision_time` ranges from 2.723 to 48.011 while the output data `exam_score` ranges from 14.731 to 94.945. Data is of the double floating point type, which is represented as <class 'numpy.float64'> in python numpy.

Question 1 continued on next page. . .

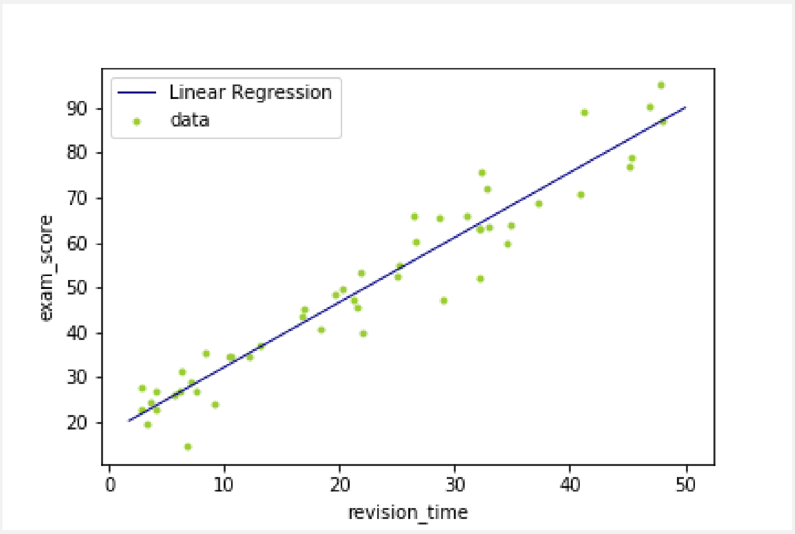
(b) (3 points) Fit a linear model to the data so that we can predict `exam_score` from `revision_time`. Report the estimated model parameters $\mathbf{w}$. Describe what the parameters represent for this 1D data. For this part, you should use the sklearn implementation of Linear Regression.
Hint: By 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 linear regression model is $y = \boldsymbol{\phi}(\mathbf{x})\mathbf{w}^T$, where $\boldsymbol{\phi}(\mathbf{x})$ is a vector which serves to $\mathcal{R}^{50\times 2}$ and $\mathbf{w}$ is a weight vector which serves to $\mathcal{R}^{1\times 2}$. The estimated model parameters $\mathbf{w}$ is equal to [17.90, 1.44]. Value 17.90 is the weight of the constant feature 1 that we have concatenated while value 1.44 is the weight of the input feature `revision_time`. The weight of the input feature `revision_time` is positive therefore `revision_time` is positive correlated to `exam_score`.

Question 1 continued on next page...

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

The following figure shows the fitted linear model with input data in the same plot.



From the figure, we see that the model is well fitted with 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 **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.array(data.iloc[:,0:2])   #x
y = np.array(data.iloc[:,2])     #y
T_x = np.linalg.pinv(x.T.dot(x)) #T_x = (x^T x)^-1
w = T_x.dot(x.T.dot(y))          #w = (x^T x)^-1 (x^T y)
```

(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.

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

One of its limitations is that it doesn't take the weight's effects into account. It might generate large coefficients to fit the outliers in order to achieve a smaller loss. More importantly, MSE expression corporates the variance and bias, where exists variance-bias trade-off.

Question 1 continued on next page. . .

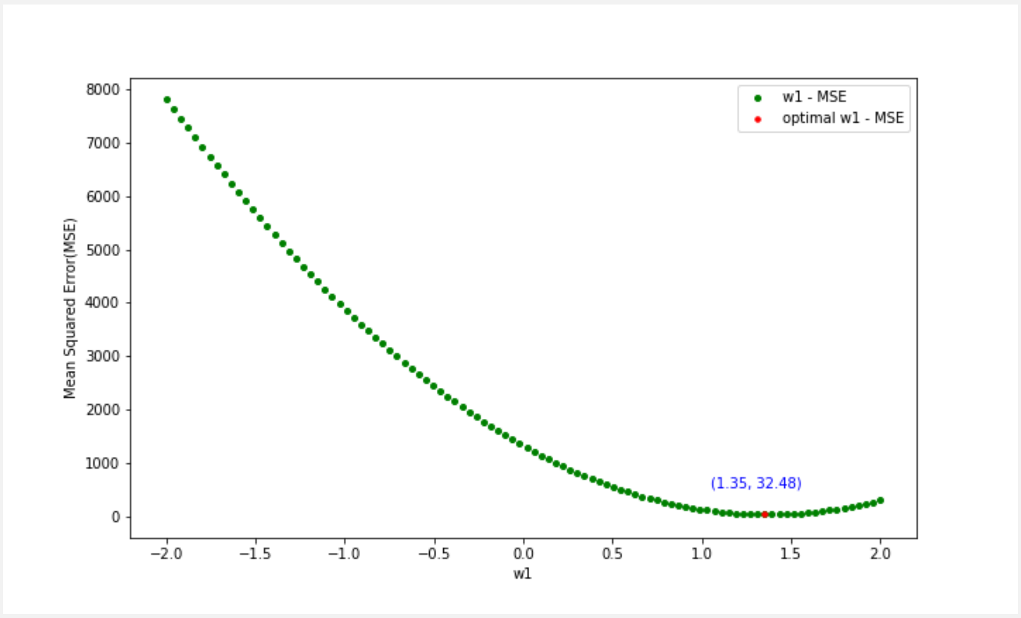
(f) (3 points) Our next step will be to evaluate the performance of the fitted models using Mean Squared Error (MSE). Report the MSE of the data in `regression_part1.csv` for your prediction of `exam_score`. You should report the MSE for the linear model fitted using sklearn and the model resulting from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

MSE for the fitted linear model using sklearn is equal to 30.985. MSE for the model using closed-form solution is also equal to 30.985. The results show almost no difference between these two results. The reason is that sklearn.LinearRegression also uses closed-form solution to estimate its parameter $\mathbf{w}$. We can check the correctness by viewing from line 520 to line 530 in sklearn/linear_model/_base.py, where uses sparse_lsqr method to estimate $\mathbf{w}$. There might be difference after 6 decimal point due to calculation precision, the ideas behind them are the same.

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 following figure shows the resulting plot which describes the mean squared error(MSE) when w1 changes from -2 to 2.



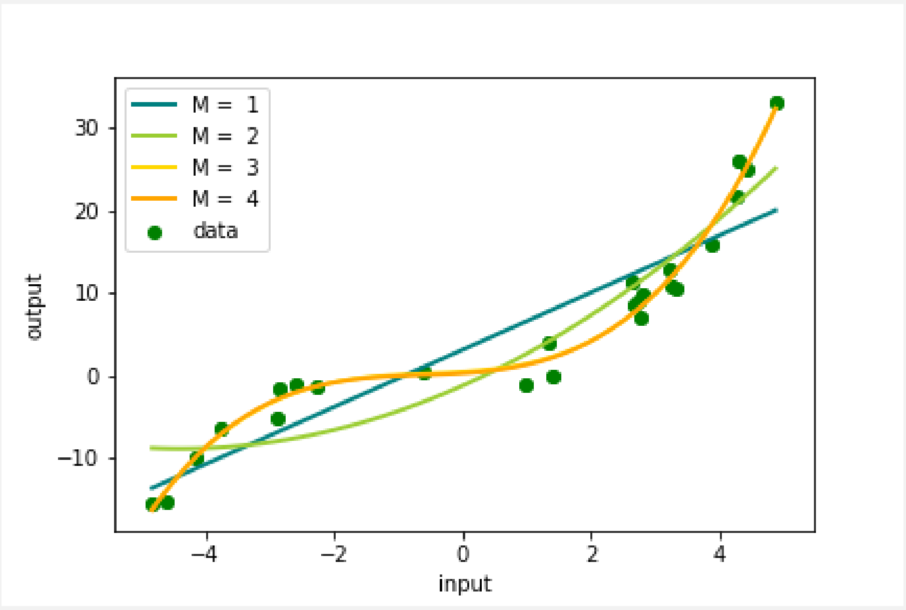
MSE reaches its minimum when w1 is equal to 1.35. The theoretical value for w1 is 1.37, which can be achieved by closed-form solution. Optimal value is the closest to theoretical value according to w1. Therefore, it's reasonable to be the optimal value for the given w1.

Question 2 : (18 total points) Nonlinear Regression

In this question we will tackle regression using basis functions.

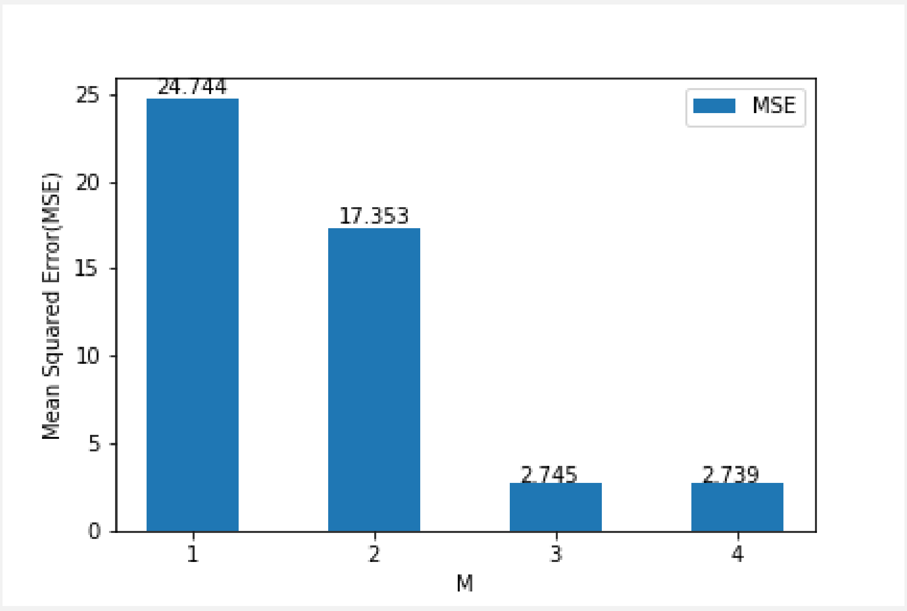
(a) (5 points) Fit four different polynomial regression models to the data by varying the degree of polynomial features used i.e. $M = 1$ to 4. For example, $M = 3$ means that $\boldsymbol{\phi}(x_i) = [1, x_i, x_i^2, x_i^3]$. Plot the resulting models on the same plot and also include the input data.
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 following plot shows four fitted models and input data.



(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 following bar plot shows the mean squared error of each of the four different polynomial regression models.



Question 2 continued on next page...

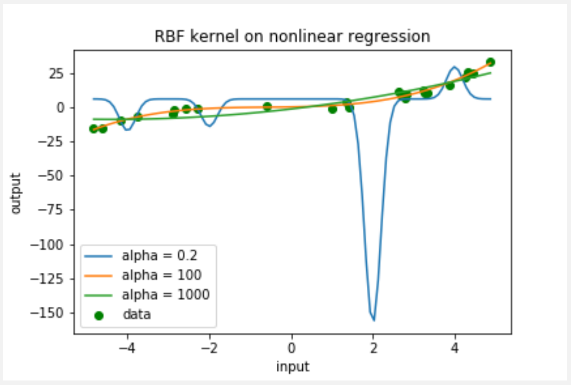
(c) (4 points) Comment on the fit and Mean Squared Error values of the $M = 3$ and $M = 4$ polynomial regression models. Do they result in the same or different performance? Based on these results, which model would you choose?

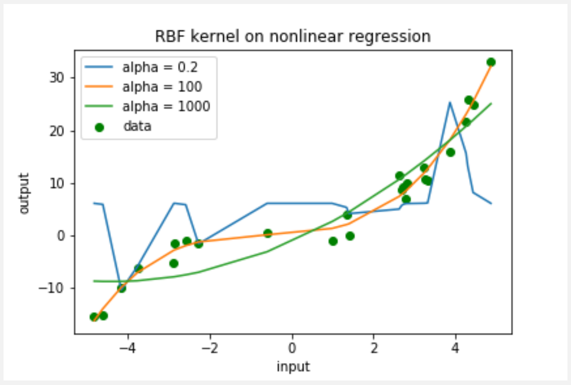
When model hyper-parameter M equals to 3 or 4, they result in almost the same performance and they fit the data well. This can be found in the plot shown in part (a) where fitted curves under these two models are overlapped. Both two models are better than models with M = 1 and M = 2. I would choose the model with M = 3. Firstly it reaches almost the best performance among all four models. Secondly, compared to the fourth model, it has fewer calculation time with the time complexity O(N^3). It really matters when numbers of input data are large. Moreover, using the model with M = 4 may cause over-fitting problem when encountering unseen data while applying the model with M = 1 and M = 2 can cause under-fitting problem.

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 following figure shows two plots. Both two plots contain 3 models based on radial basis function. The difference is that left plot is fitted with 200 points while plot on the right is fitted with only input data.





We treat α as the standard deviation. From the plot we can figure out that a larger value of α makes the curve smoother. A smaller value of α makes the curve sharper.

Question 3 : (26 total points) Decision Trees

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

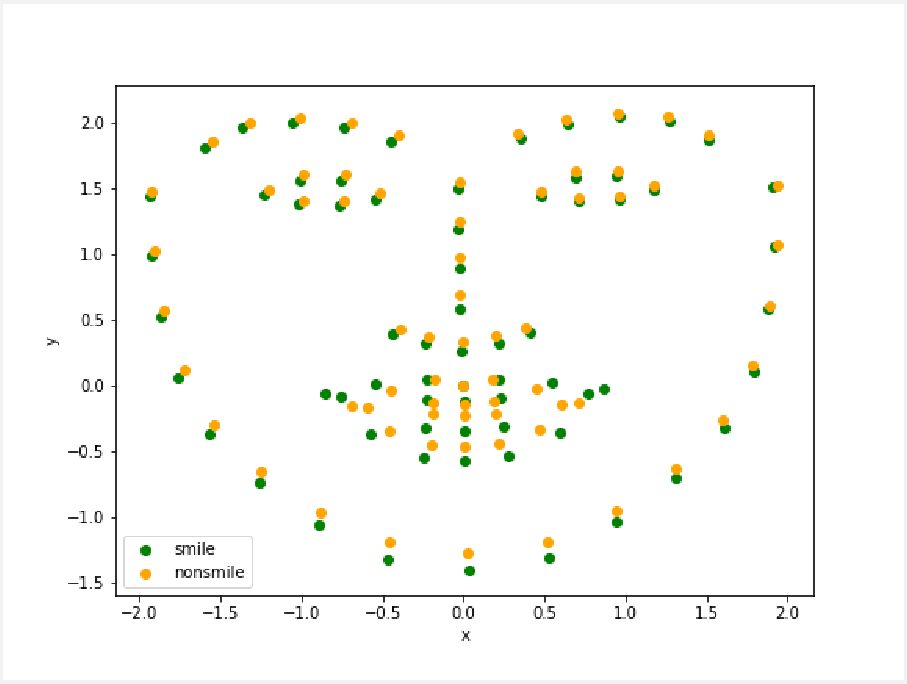
(a) (4 points) Load the data, taking care to separate the target binary class label we want to predict, `smiling`, from the input attributes. Summarise the main properties of both the training and test splits.

There are 4800 numbers of training data and 1200 numbers of test data in total. The ratio of training set to test set is 4 to 1. The number of features is 136 for both splits. Data type of each feature is double precision floating point. It is represented as "numpy.float64" in python numpy array. Label 'smiling' for prediction is of binary data type. Feature (Coordinate) $x_i, i \in 0, 1, ..., D$ is in the range of -3.98 to 3.92 while feature (Coordinate) $y_i, i \in 0, 1, ..., D$ is in the range of -3.52 to 3.58 according to training set. Feature (Coordinate) $x_i, i \in 0, 1, ..., D$ is in the range of -3.85 to 3.90 while feature (Coordinate) $y_i, i \in 0, 1, ..., D$ is in the range of -2.62 to 3.64 according to test set. Here D is equal to 67.

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 following figure shows the scatter plot of the average location for each 2D coordinate for both smiling face(green colored) and non-smiling face(yellow colored).



Main difference between two sets of points is that points with smiling $= 0$ have a wider horizontal and vertical range across the mouth.

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 criteria of deciding the impurity of the node is Gini impurity. Gini impurity is represented by $G_i = 1 - \sum_{k=1}^{M} p_{i,k}^2$. Entropy impurity is represented by $H_i = 1 - \sum_{k=1}^{M} p_{i,k} * log_2 p_{i,k}$. One obvious advantage is that entropy calculation needs logarithm operation, which costs more time than multiplication operation that is required by Gini impurity. Besides, we know intuitively that it tends to choose the most frequent class of at the branches.

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 potential problem which small values of maximum depth of the tree may occur is that the impurity is high at the top nodes of the tree. It will result in under-fitting problem. Maximum depth with unsuitable large values will cause over-fitting problems. Also, if numbers of data is large, it will spend more time on calculation. Expected time complexity is O(nlgn) since we choose lg(n) features in our tree. Time complexity will become $O(n^2)$ if we add more numbers of features to the branches until it reaches the number of maximum input features.

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 shows the training accuracy and test accuracy under different maximum depth.

maximum depth	training acc(%)	test acc(%)
2	79.48	78.17
8	93.35	84.08
20	100.00	81.50

The second model with the maximum depth of 8 preforms best. Since value 8 is the closet to expected maximum depth lg(n) where n is equal to 136. ln(136) is equal to 7.09.

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`, of which the importance is equal to 0.3304, 0.0900 and 0.0883 in order respectively. The attribute with the highest importance is `x50`. It makes sense since top of the upper lip, whose horizontal value is represented by `x50` will extend when one person is smiling. It is common sense although the disparity is not obvious in the figure shown in part(b).

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 is that more than half of the coefficients(feature importance) are equal to zero(71/136) according to the reported `feature_importances_`. It means that those features are not sensitive to label prediction, which is redundant to classifier.

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?

Classification accuracy for each of the four different models are presented in the table below with the threshold 0.5.

Algorithm	Accuracy(%)
alg_1	61.6
alg_2	55.0
alg_3	32.0
alg_4	32.9

"alg_1" is the best model according to the metrics. However, one limitation is that threshold is fixed. We can try to use different thresholds to improve accuracy. Moreover, we can average the accuracies with selected thresholds.

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 roc curve(AUC) for each model is reported as below.

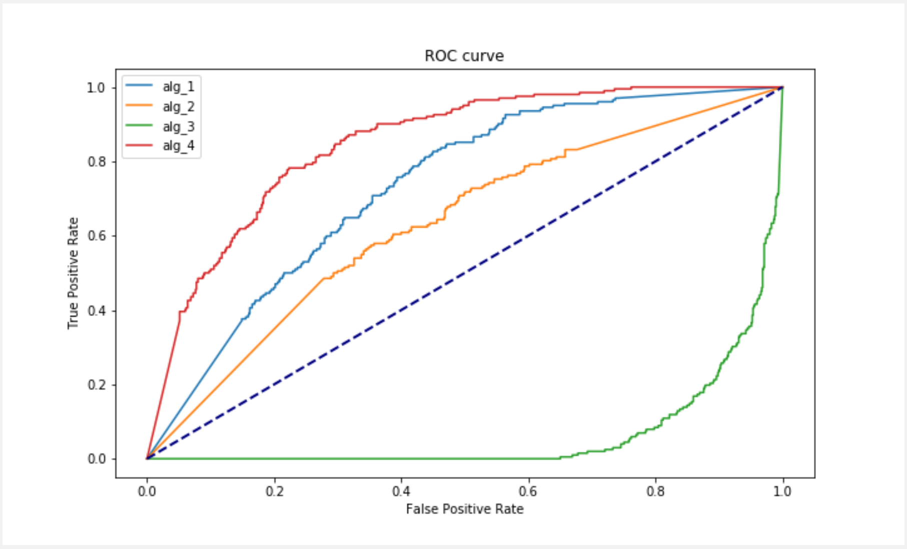
Algorithm	AUC
alg_1	0.732
alg_2	0.632
alg_3	0.064
alg_4	0.847

The model with the best AUC is alg_4 but it doesn't have the best accuracy. In question (a), the threshold is set to 0.5. However, AUC calculates the average accuracies of all threshold value from 0 to 1. We speak of accuracy as the condition where threshold is equal to 0.5 but it only represents one point on AUC curve.

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 is shown below.



Since from the ROC curve, we can figure out that alg_3 is totally an inverse model since its AUC value is less than 0.5 and even close to 0. So we change the result data of alg_3 from y to 1-y. The AUC value will become 0.936, reaching the best AUC among all four models.