IAML – INFR10069 (LEVEL 10): Assignment #1

s1894401

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 training inputs - each with a single attribute named `revision_time`, which is a 64-bit floating point number - and 50 training outputs. The training output variable is called `exam_score` and it is also a 64-bit floating point number. Across the dataset, `exam_score` ranges from a minimum value of 14.731 to a maximum value of 94.945 while `revision_time` ranges from a minimum value of 2.723 to a maximum value of 48.011.

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

$$\boldsymbol{w} = \begin{bmatrix} 17.8977 \\ 1.4411 \end{bmatrix}$$

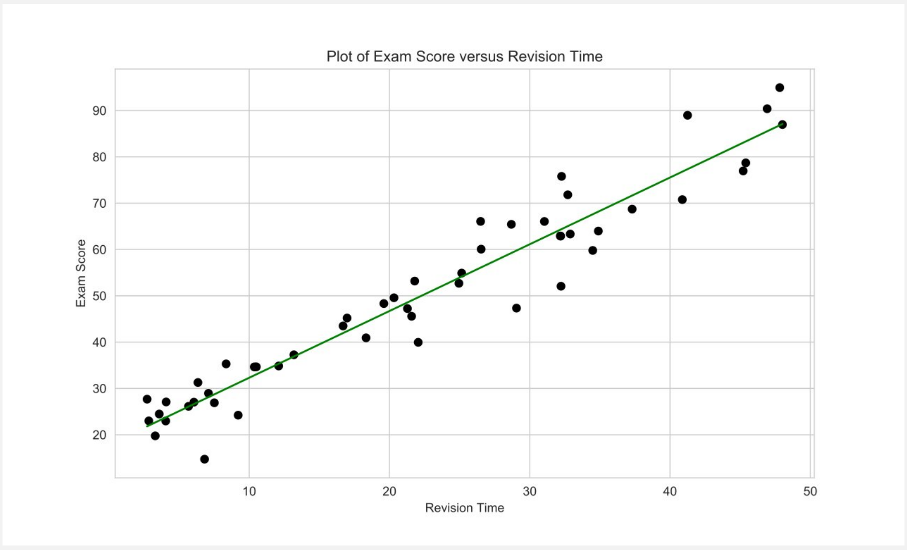
with values given to 4 decimal places.

$\boldsymbol{w}[0]$ represents the score you get if you do not study at all (when `revision_time` is 0) and it is also the intercept of the line at the ordinate axis.

$\boldsymbol{w}[1]$ represents the gradient or slope of the regression line.

Question 1 continued on next page...

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



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.*

```
phi = np.array([[1.,x] for [x,y] in regression_part1[:].values])
y = np.array([y for [x,y] in regression_part1[:].values])
phi_T_phi_inv = np.linalg.inv(phi.T.dot(phi))
w = (phi_T_phi_inv.dot(phi.T)).dot(y)
```

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.

One limitation of MSE is that it is prone (sensitive) to outliers. The expression for MSE is:

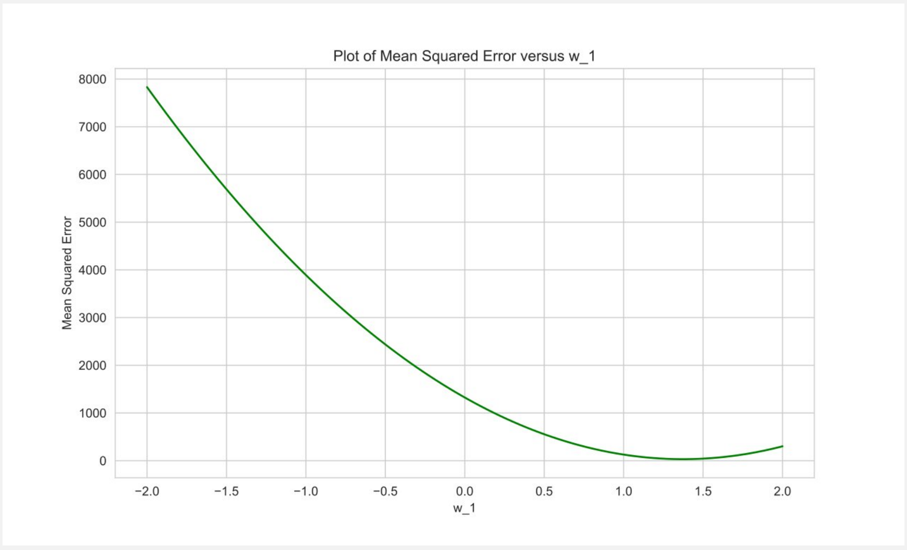
$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

(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 30.985473 (to 6 d.p.) while that for the closed-form solution is also 30.985473 (to 6 d.p.). The fact that both values are the same (up to the 12th decimal place) suggests that sklearn uses the closed-form solution in its implementation of the **LinearRegression** class methods.

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)`.

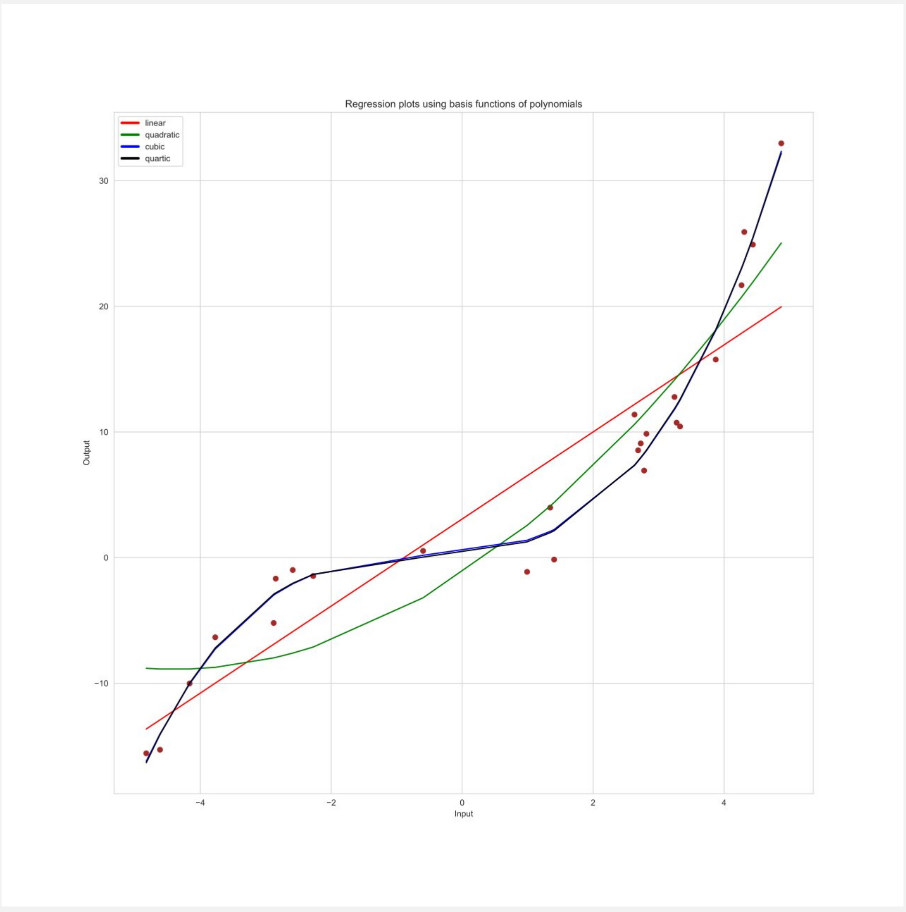


The resulting graph is a parabola whose minimum value is 32.48 (to 2 d.p.) along the y-axis, corresponding to a value of 1.35 (to 2 d.p.) along the x-axis. The value of the x-coordinate at this minimum gives us the optimal value of $\mathbf{w}[1]$ when $\mathbf{w} = [20, w_1]$. Yes this is the expected value given that for the actual optimal value of $\mathbf{w}[0]$ (17.9 to 1 d.p.), the associated value of $\mathbf{w}[1]$ is 1.44 (to 2 d.p.) which is in the neighbourhood of 1.35.

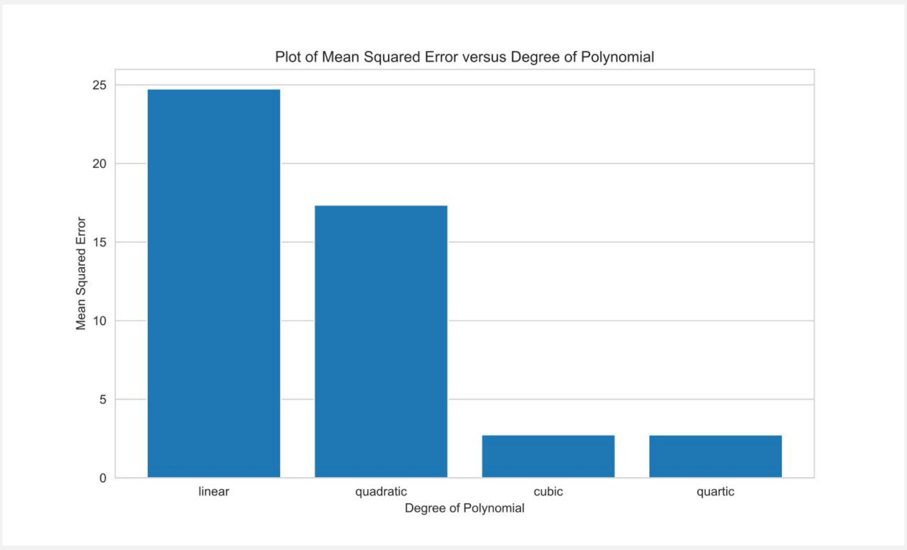
Question 2 : (18 total points) Nonlinear Regression

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.



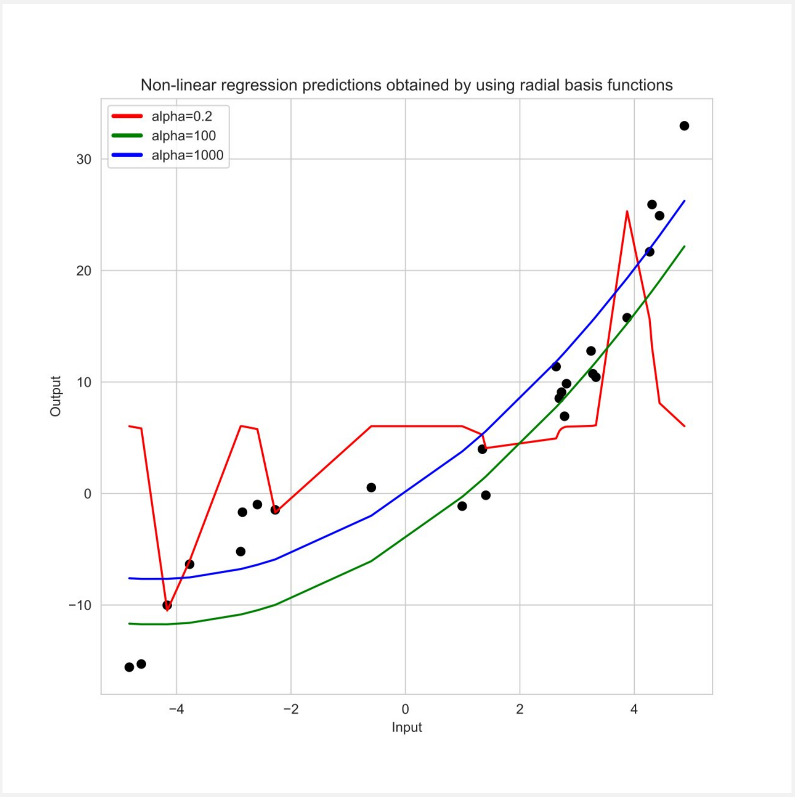
Question 2 continued on next page…

(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?

The Mean Squared Errors (MSE) of $M = 4$ and $M = 3$ differ only slightly (by about 10^{-2}). Even though the MSE of $M = 4$ on the training data is smaller than that of $M = 3$, $M = 4$ suffers from overfitting as it captures the noise in the training data. This means $M = 3$ will be able to generalise better than $M = 4$ on unseen data. I would therefore choose model $M = 3$.

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.



Question d: As seen from the graph, for very small values of α, the predictions from the RBFs suffer from underfitting. Also, for very large values of α, the RBFs suffer from underfitting too. When the value of α lies between 15 and 35, the RBFs fit the data very well.

Question 3 : (26 total points) Decision Trees

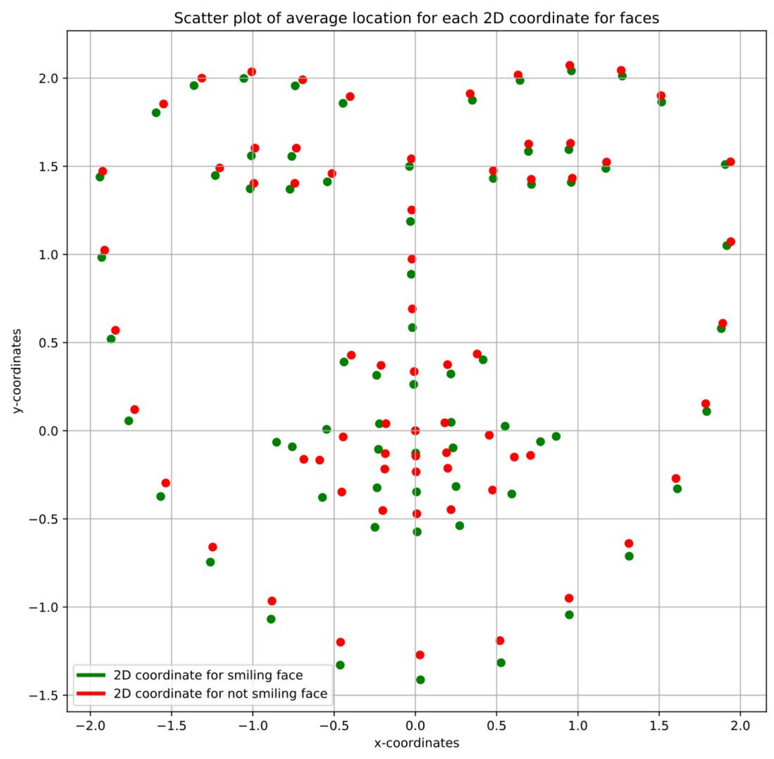
In this question we will train a classifier to predict if a person is smiling or not.

(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 and testing inputs are row vectors each of which have 136 attributes. Each attribute is stored as a 64-bit floating point number.
The training dataset has 4800 inputs. The output vector for the training set is a column vector comprising 4800 binary labels with each label stored as a 64-bit integer.
The testing dataset has 1200 inputs. The output vector for the testing set is a column vector comprising 1200 binary labels with each label stored as a 64-bit integer.

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 data points for smiling faces are further away from each other (more spread out) around the lips and chin as compared to those for the not smiling faces.

(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?

Sklearn uses the gini index to measure the purity of a node. The gini index is faster to compute as opposed to entropy as entropy makes use of the logarithm for its computation.

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.

Low values of `max_depth` lead to underfitting.
High values of `max_depth` will lead to:
- complex data models that are difficult to interpret and expensive to compute.
- models that overfit to the training data, thereby preventing them from generalising on testing data.

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.

Tree with `max_depth` of 2 : Training accuracy is 79.48% and testing accuracy is 78.17%

Tree with `max_depth` of 8: Training accuracy is 93.35% and testing accuracy is 84.08%

Tree with `max_depth` of 20: Training accuracy is 100.00% and testing accuracy is 81.58%.

All values are given to 2 decimal places.

The tree model with a `max_depth` of 8 is the best because it has the best accuracy on the testing data - which means it is better at generalising on unseen data as compared to the the other 2 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`.

Most important attributes in descending order of Gini importance:
x50, **x48**, **y29**
The mean value (for the training data) of the 2D coordinate for the points containing the most important attribute on a smiling face is (-0.22, 0.04) and that for a non smiling face is (-0.18, 0.04). From the scatter plot in Question (b), we see that these correspond to a point on the upper lip of a face. It makes sense that this is the most important feature as we notice that the lips dilate when a person smiles.

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.

Yes there are. A point on the human face is naturally represented as a 3D coordinate in the real world but it is represented as a 2D coordinate in our dataset. This representation leads to loss of information as we have ignored one of the dimensions which could have potentially play a huge part (as in providing us with features of higher importance) in the classification task.

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?

Model 1 is the best according to this metric as seen from the classification accuracy of the algorithms:
`alg_1` = 61.6%, `alg_2` = 55.0%, `alg_3` = 32.1%, `alg_4` = 32.9%

One limitation of this measure is that it does not work well with a class-imbalanced dataset. It can be improved by using a dataset with the same number of data points for each class during training.

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.*

AUC for the models:

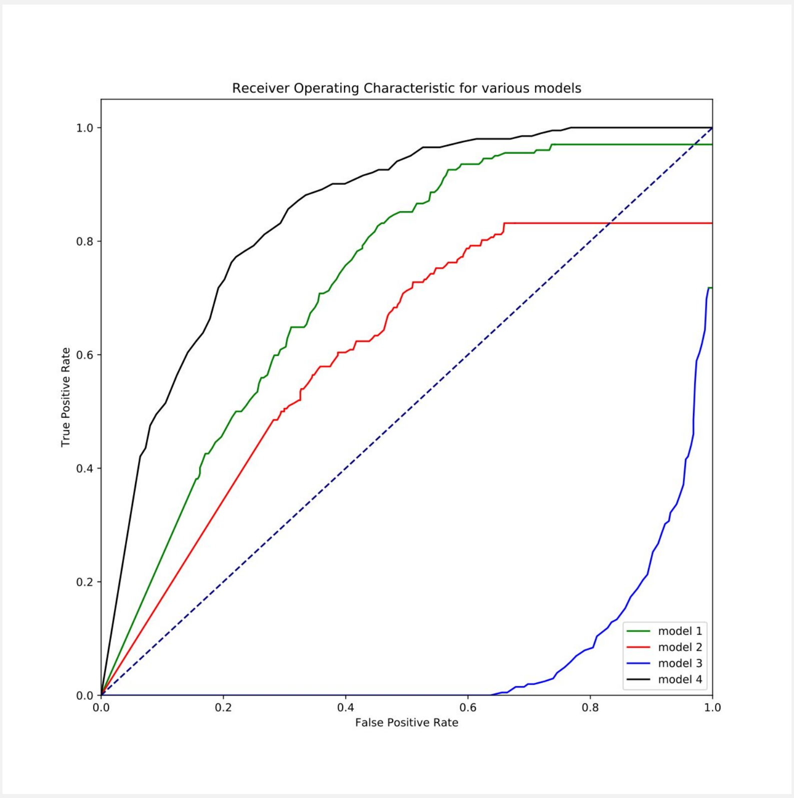
Model 1 : 0.6799
Model 2 : 0.5997
Model 3 : 0.2011
Model 4 : 0.5796

All the answers are given to 4 decimal places.

Yes the model with the best AUC score is also the one with the highest accuracy.

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 `alg_3` is worse than that of a model with random performance. Though setting a very low threshold (one very close to 0) will increase the true positive rate of `alg_3`, its false positive rate would still be very high (close to 1.0). Thus, nothing can be done to improve the performance of `alg_3` without having to retrain the model.