IAML – INFR10069 (LEVEL 10): Assignment #1

s1803764

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.

Data properties of the 'regression_part1.csv' dataset:

Dataset size: 50 records
Dataset shape: (50,2)
Data types: all attribute values of type float64

ATTRIBUTES:

revision_time (in hours)

Possible data range: $0 \leq \mathbf{x}_i \leq \alpha$, where α represents the available hours of study
Dataset data range: $2.723 \leq \mathbf{x}_i \leq 48.011$
Mean: $\mu_x \approx 22.22$
Standard deviation: $\sigma_x \approx 13.99$

exam_score (in %)

Possible data range: $0 \leq \mathbf{y}_i \leq 100$
Dataset data range: $14.731 \leq \mathbf{y}_i \leq 94.945$
Mean: $\mu_y \approx 49.92$
Standard deviation: $\sigma_y \approx 20.93$

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

Fitting a linear model to predict exam_score from revision_time

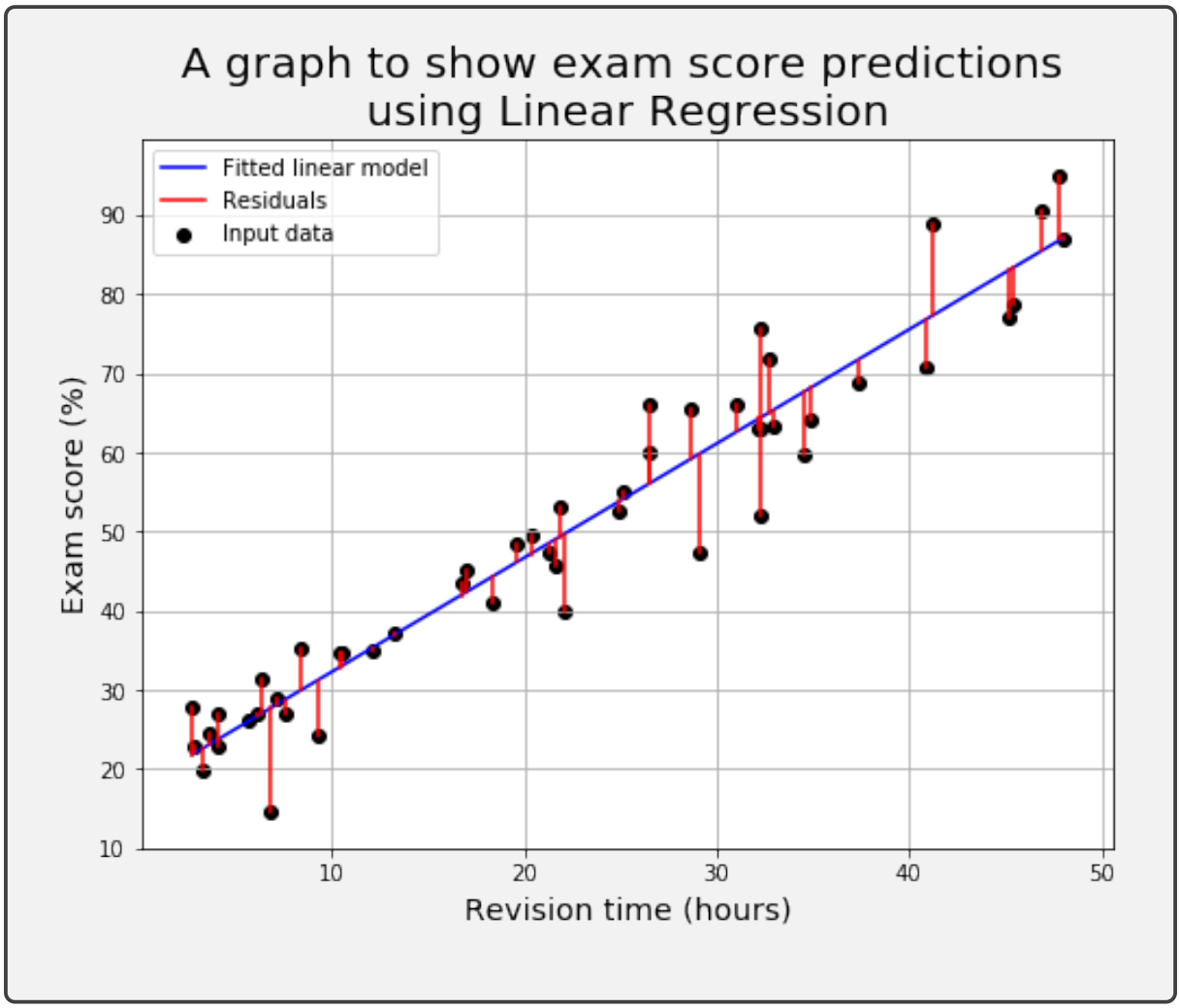
Linear model:
Coefficient = 1.44114091
Intercept = 17.89768025835017

Therefore:
$\boldsymbol{w} \approx [17.898, 1.441]$

Representation of $\boldsymbol{w}$:
$\boldsymbol{w_0}$ represents the y-intercept of the model and thus denotes the minimum estimated possible exam score.
$\boldsymbol{w_1}$ represents the gradient of the model and thus quantifies the direct proportionality between the student's hours of study and their exam score.

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 $\mathbf{w}$ *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.*

Closed-form solution for fitting a linear regression model with numpy

Given our analytical solution for calculating $\hat{\mathbf{w}}$:

$$\hat{\mathbf{w}} = (\mathbf{\Phi^T} \cdot \mathbf{\Phi})^{-1} \cdot \mathbf{\Phi^T} \cdot \mathbf{Y}$$

where $(\mathbf{\Phi^T} \cdot \mathbf{\Phi})^{-1} \cdot \mathbf{\Phi^T}$ is the pseudo-inverse of $\mathbf{\Phi}$

We can easily transfer this into code:

```
pseudoInversePhi = np.matmul(np.matrix(np.matmul(phi.transpose(), phi)).I, phi.transpose())

w_hat = np.matmul(pseudoInversePhi, Y)

print(w_hat)
```

```
[[17.89768026]
 [ 1.44114091]]
```

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.

Limitations of the Mean Squared Error (MSE) metric

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

One of the disadvantages of the MSE metric is that it is very sensitive to outliers. This is evident in the equation as shown above, where for each value of x_i the error value (difference in the predicted and actual value) of y_i is squared and the average error value is calculated. This squaring of error values ultimately magnifies the large errors (caused by outliers) and minimizes the small errors. Inevitably meaning a single outlier in our dataset could skew our analysis greatly, due to the fact that we could get the same MSE for a predictor with one large error as a predictor with lots of small errors.

The best way to solve such a problem would be through either identifying outliers in our dataset through visualization or using different error metrics which are more robust against outliers (eg. Median Absolute Deviation).

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.

Implementation evaluation using MSE

$MSE_{\text{sklearn}} \approx 30.985$

$MSE_{\text{closed-form}} \approx 30.985$

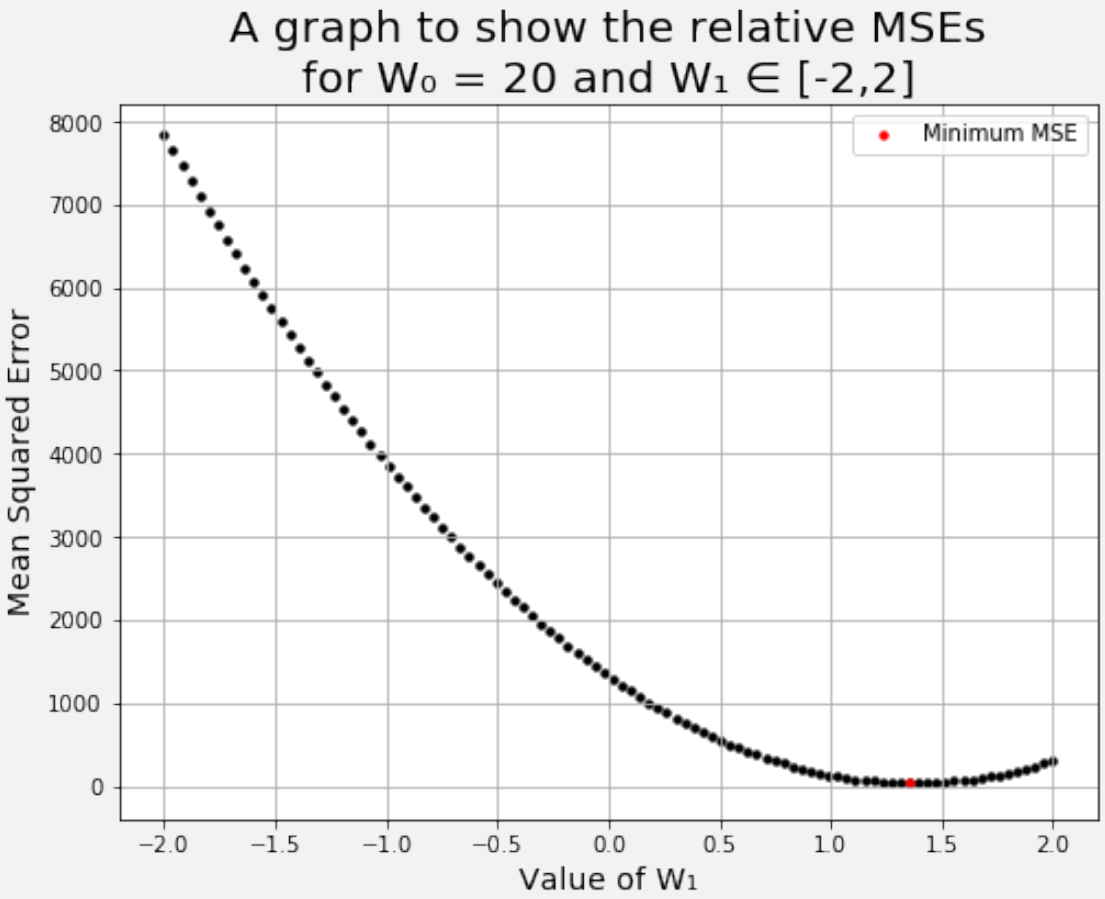
Differences in performance:

$MSE_{\text{sklearn}} \approx MSE_{\text{closed-form}} + 1.421 * 10^{-14}$

The MSEs of these implementations were incredibly close. Given that both of these models were calculated on the exact same dataset, we can see that the solution of our **closed-form** implementation was a tiny bit more accurate than that of our sklearn implementation.

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)`.



This plot seems to take the form of a parabola, where the MSE increases exponentially either side of the global minimum MSE (illustrated by the red point). This shows just how important the decimal accuracy of our $\mathbf{w}$ value really is, as a small difference in this can result in a large difference in accuracy.

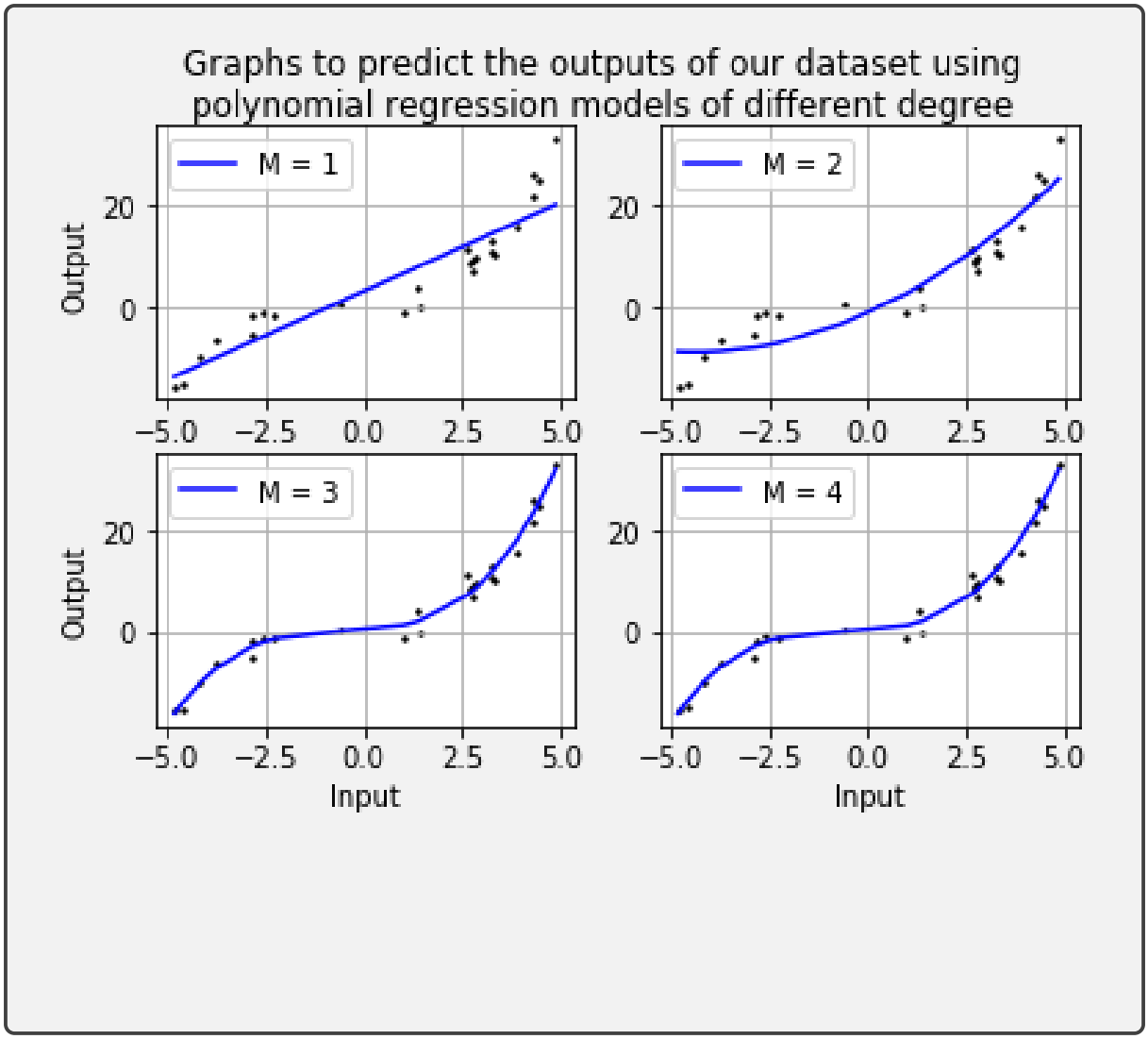
$\mathbf{MSE_{min}} \approx 32.481$, where $\mathbf{w_1} \approx 1.354$

This value of $\mathbf{w_1}$ is expected. We knew it had to be less than that of the optimum found in previous implementations (where $\mathbf{w_1} \approx 1.441$). This is due to the fact that we have set $\mathbf{w_0} = 20$, which is evidently greater than its optimum (where $\mathbf{w_0} \approx 17.898$).

Question 2 : (18 total points) Nonlinear Regression

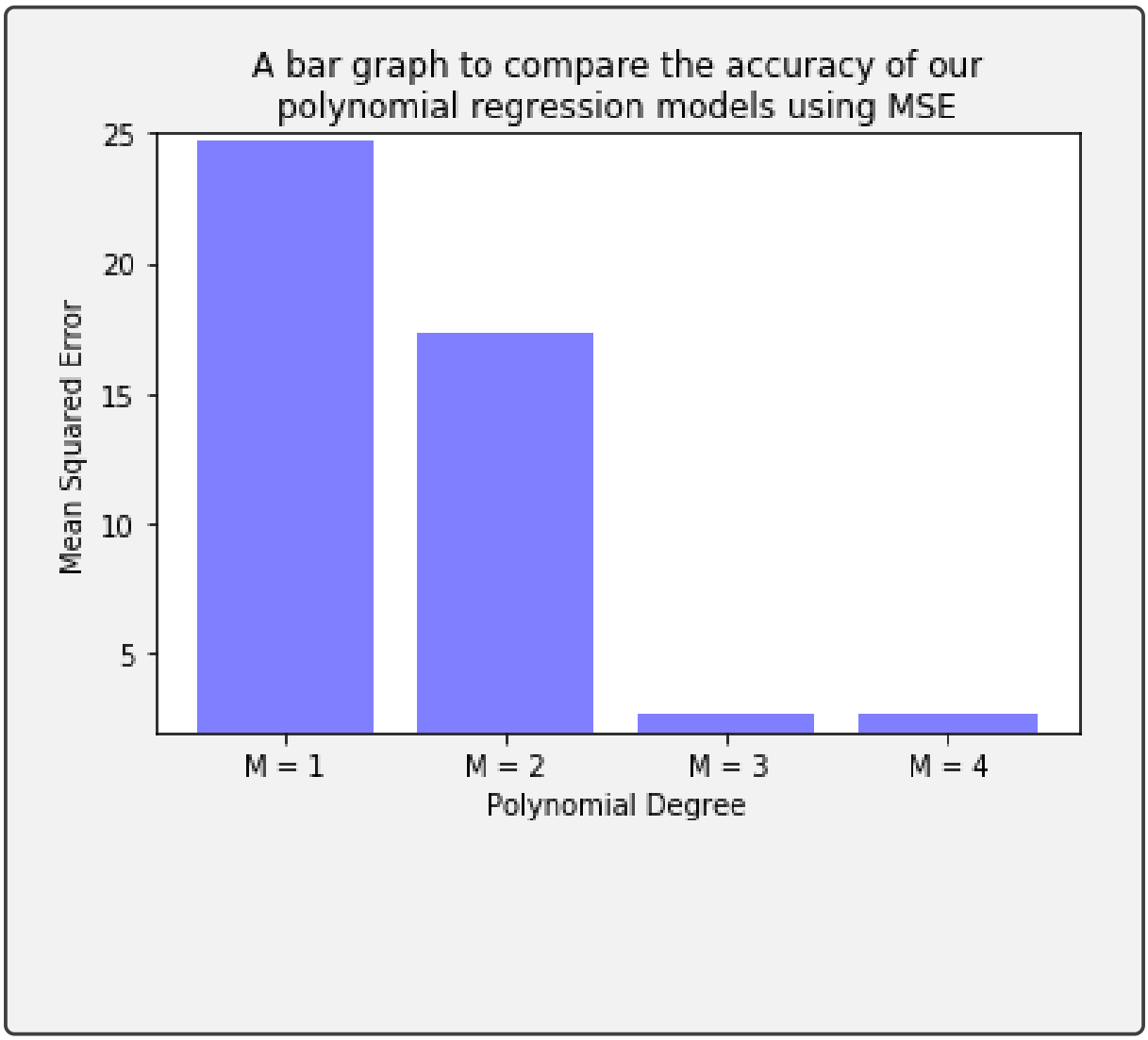
In this question we will tackle regression using basis functions.

(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`.



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.



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?

Comparing the M=3 and M=4 polynomial regression models

$MSE_{M=3} \approx 2.745$
$MSE_{M=4} \approx 2.739$

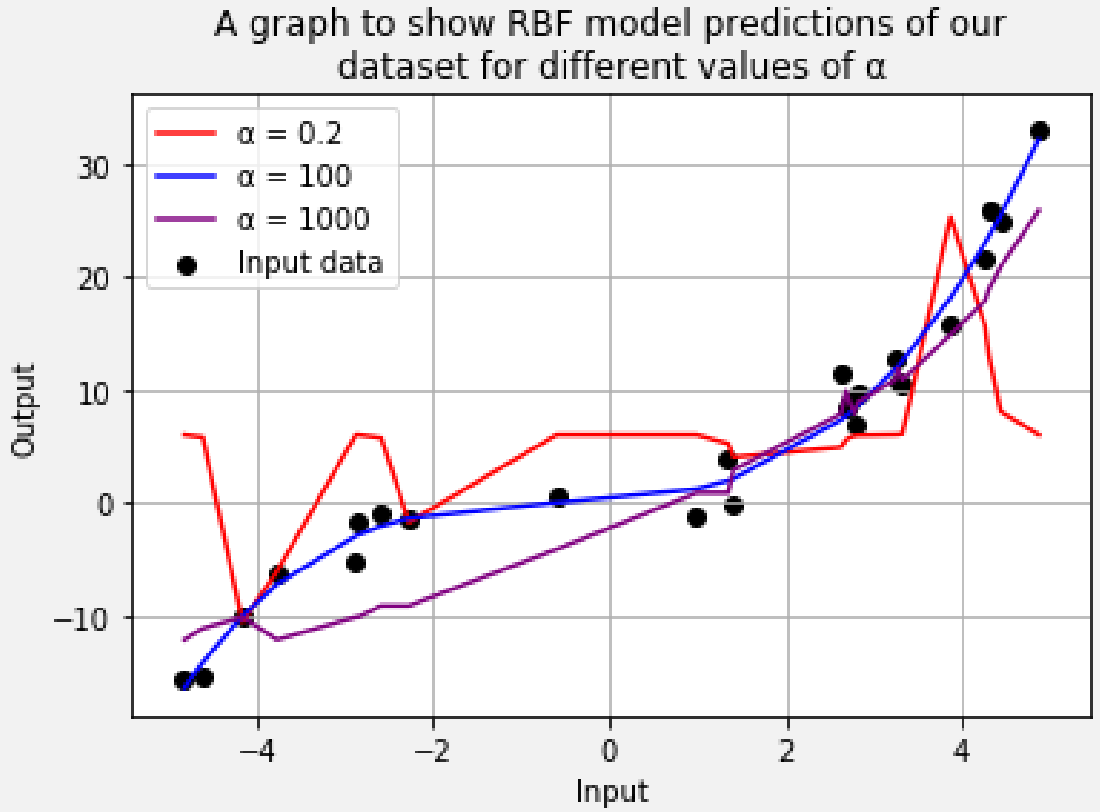
Such that:
$MSE_{M=3} \approx MSE_{M=4} + 5.846 * 10^{-3}$

It is therefore evident that the M=4 model performs slightly better than the M=3 model on this given dataset. However we must be wary of overfitting these models to our training data. So before deciding on which model to use it would be useful to test and compare the performance of these models on a large unseen dataset.

But since we are unable to test these models any further if I had to choose a model given it's MSE performance metric on this dataset alone I would choose the **M=4** model.

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.



Impact of α:

As illustrated on this figure it is evident that the RBF kernel with $\alpha = 100$ creates the best classifier due to it's smooth model and close fit to all the data points.

The RBF kernel with $\alpha = 0.2$ seems to overfit the data in a dramatic way with a very "spikey" model that does not fit well to a large majority of the data points. In contrast, the RBF kernel with $\alpha = 1000$ seems to underfit the data with a relatively straight model that does not fit well to a smaller majority of the data points.

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.

INPUT DATA:

Data properties of the 'faces_train_data.csv' dataset:

Dataset size: 4800 records
Dataset shape: (4800, 136)
Data types: all attribute values of type float64

Data properties of the 'faces_test_data.csv' dataset:

Dataset size: 1200 records
Dataset shape: (1200, 136)
Data types: all attribute values of type float64

OUTPUT DATA:

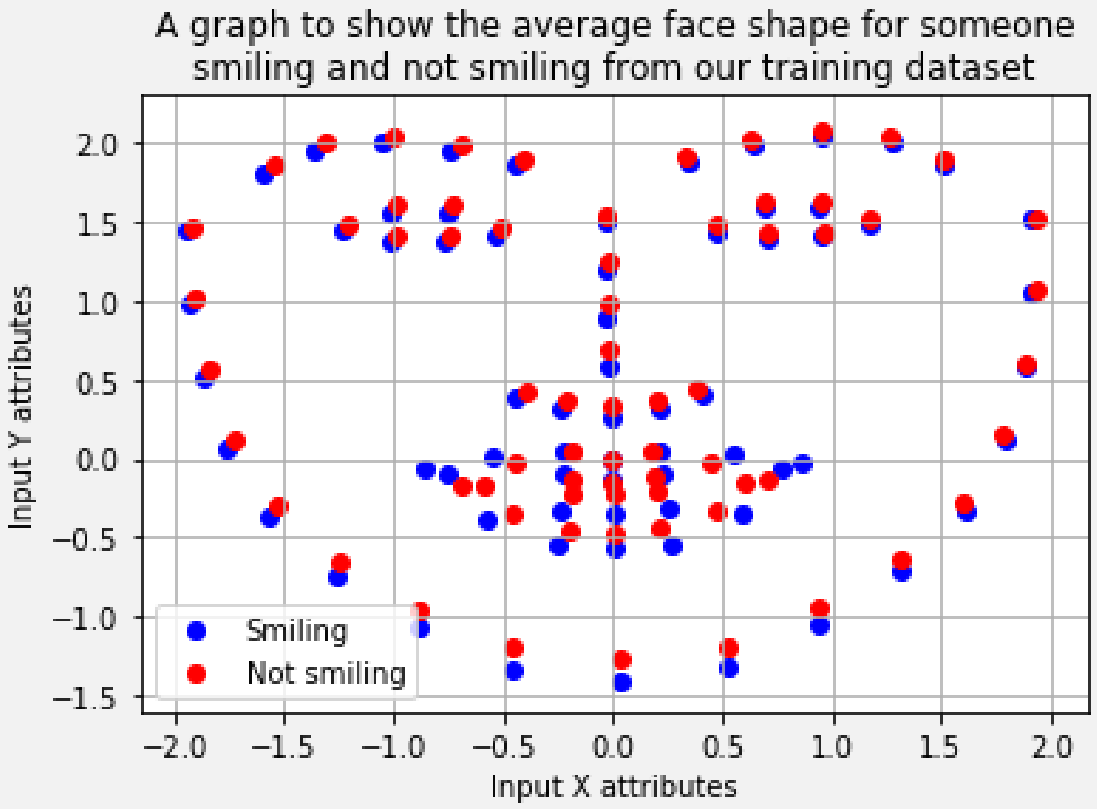
Data properties of the "smiling" output attribute:

Data type: attribute value of type int64

This "smiling" attribute is a binary value which is used to label whether a given piece of data represents someone smiling or not.

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.



Differences between the two sets of points:

We can see that for the average smiling face the mouth shape is significantly wider, the chin is quite a lot lower, and overall the face is slightly wider than that of the non-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?

Decision Tree split quality evaluation measures:

By default the *DecisionTreeClassifier* in sklearn uses the Gini Impurity metric to measure the purity of a node for classification.

The Gini Impurity and Entropy metrics both measure the impurity of a node, and are thereby used as criterion for calculating information gain. Decision Tree algorithms use this information gain to choose which node to split on.

The internal working of both these measures are very similar, the only real advantage of using Gini Impurity rather than Entropy is that it requires less computational power. This difference in computational power can make a huge difference in computational time when working with big decision trees.

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.

Impact of the 'max_depth' parameter on Decision Tree classifiers:

This 'max_depth' parameter denotes the maximum allowed depth of the Decision Tree. If no value is given, then nodes will be expanded until all leaves are pure or until all leaves contain less samples than the specified 'min_samples_split' parameter (this denotes the minimum number of samples required to split an internal node) which is greater than or equal to 2.

The larger the depth of the Decision Tree thereby implies a larger amount of splits to be computed.

Problem for small values of max_depth:

1. Smaller tree depths ultimately denote less splits on the samples, and are thereby more likely to be less accurate/underfitted to our data.

Problems for large values of max_depth:

1. Larger tree depths ultimately denote more splits on the samples and thereby more computational power.
2. If our tree depth is too large and 'min_samples_split' is too small we may overfit to our data as this may create singleton subsets.

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.*

Evaluating the accuracy of Decision Tree classifiers for different values of 'max_depth':

max_depth = 2:
Training accuracy ≈ 79.48%
Testing accuracy ≈ 78.17%

max_depth = 8:
Training accuracy ≈ 93.35%
Testing accuracy ≈ 84.08%

max_depth = 20:
Training accuracy = 100%
Testing accuracy ≈ 81.58%

Choosing the best model:

Given the training and testing accuracies of these models, the model with **max_depth = 8** is the best. This is evident as it has the highest testing accuracy.

Testing accuracy is the most important measure here as it illustrates how well a given model reacts to unseen data. Training accuracy is trivial in comparison as this does not tell us whether our model has been overfitted to noise in the training data.

Although the model where max_depth = 20 has a perfect training accuracy (100%) it has a worse testing accuracy than the max_depth = 8 model which implies it must have been overfitted.

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`.

Top 3 most important attributes according to Gini importance:

1. **x50**
2. **y48**
3. **y29**

The **x50** attribute represents one of the points on the top of the face's mouth. Therefore it is not surprising this was given the highest Gini importance as obviously any points on the mouth would be the most significant for recognizing a smile.

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.

Limitations of the current choice of input attributes:

- They are not very detailed (limited number of 2D point location attributes) and therefore do not account for other features associated with smiling such as puffy cheeks or dimples.
- They do not account for faces pointed in different angles.

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 of our 4 models:

$$\textbf{Accuracy} = \frac{\textbf{TP+TN}}{\textbf{TP+TN+FP+FN}}$$

alg_1 = 61.6%
alg_2 = 55%
alg_3 = 32.1%
alg_4 = 32.9%

According to this metric we can see that **alg_1** is the best model.

Limitations of this method for computing accuracy:

This accuracy metric just gives us a basic overall measure of the "goodness" of our classifier, it does not tell us the relative accuracies for classifying each class. Thereby this metric is not very useful when we want to know the strong and weak areas of our given classifier, and can become very misleading when you have imbalanced classes.

To improve this method for computing accuracy without changing our metric I would optimize our threshold value for binary classification. In order to do this I would iterate through threshold values in the range [0,1] and choose the threshold that produces the overall best accuracy for all the models.

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 evaluations of our models:

alg_1 ≈ 67.99%
Where TPR ≈ 78.71% & FPR ≈ 42.73%

alg_2 ≈ 59.97%
Where TPR ≈ 68.31% & FPR ≈ 48.37%

alg_3 ≈ 20.11%
Where TPR = 0% & FPR ≈ 59.77%

alg_4 ≈ 57.96%
Where TPR ≈ 100% & FPR ≈ 84.09%

So yes, **alg_1** has both the best accuracy and the best AUC throughout all the models.

Although alg_4 has a perfect TPR it has a very poor FPR (almost double that of alg_1) which greatly brings down it's AUC score, whereas, alg_1 has the best FPR and the second best TPR.

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.

Your Answer Here