Machine Learning

SAMPLE EXAM # 1

Total Time: 75 minutes

Name: ______________________

UID: ______________________

Question	***Point***	***Grade***
1	***25***	
2	***25***	
3	***20***	
4	***30***	
Total	***100***	

Person Sitting To Your Left:

Person Sitting To Your Right:

1. (25 points) Concepts.

For each of the following questions, please provide your answer and an explanation/justification. Please answer the following questions

a) (10 points) $Gini = \sum_{k=1}^{K} \quad \widehat{p_{mk}}(1 - \widehat{p_{mk}})$, where K is the number of classes, and $\widehat{p_{mk}}$ represents the proportion of the training observations in the mth region that are from the kth class.

   Please describe how classification trees use this information to train a decision tree.

b) (15 points) Please write which algorithm: Linear Regression, Logistic Regression, Boosting, or None is associated with each of the following Loss Functions

There are no typos.

i) $L(y, f(x)) = -\sum_{i=1}^{N} \quad (y_i \, log \, (f(x_i)) + (1 - y_i) \, log \, (1 - f(x_i))$

ii) $L(y, f(x)) = \quad \sum_{i=1}^{N} \quad (0, -y_i f(x_i))$

iii) $L(y, f(x)) = \quad \sum_{i=1}^{N} \quad (y_i - f(x_i))^2$

iv) $L(y, f(x)) = \sum_{i=1}^{N} \quad e^{(-y_i f(x_i))}$

v) $L(y, f(x)) = \quad \sum_{i=1}^{N} \quad y_i + f(x_i)$

2. (25 points) Logistic Regression

Assume you have collected data for a group of students with variables X1 = Amount of Hunger X2 = Is the Restaurant Busy (Yes or No), and X3 = Money Spent this week on food and outcome Y = whether to eat at a particular restaurant (Yes or No).

With this initial training data, you want to train a logistic regression model to predict Y

a) Amount of Hunger (X1): [1, 2, 3, 4, 5]
b) Busy (X2): [1, 0, 1, 0, 1]
c) Spent (X3): [90, 80, 70, 60, 50]
d) Eat Out (Y): [0, 0, 1, 1, 1]

a) (10 points) If the model trains to coefficients You fit a logistic regression model and produce estimated coefficients β_0 = -7, β_1 = 0.5 and β_2 = 1.5 β_3 = 0.5.

Given the probability of an event is calculated as

$$p(x) = \frac{e^{\beta 0\, +\, \beta 1\ x_1 + \beta 2\, x_2}}{1 +\ e^{\beta 0\, +\, \beta 1\ x_1 + \beta 2\, x_2}}$$

Which can also be calculated as

$$log\left(\frac{p(x)}{1 - p(x)}\right) = \beta 0\ +\ \beta 1\ x_1 + \beta 2\, x_2\ +\ \beta 3\, x_3$$

If restaurants are Busy, and you have spent $85 already this week, how much hunger is necessary to choose to eat out at a restaurant, with probability of 80%? Please just give the proper formulation.

This Page Intentionally Left Blank

b) (15 points) Imagine you have a linear model with 100 input features of which 10 are highly informative and 90 are noisy, uninformative. Assume all features have a range between -1 and 1. Please indicate whether each of the following statements is true or false:

- L2 regularization may cause the model to learn a moderate weight (statistically significant) for some non-informative features

- L2 regularization will encourage many of the non-informative weights to be nearly (but not exactly) 0 (and non-statistically significant).

- L1 regularization will encourage many of the non-informative weights to be nearly (but not exactly) 0 (and non-statistically significant)

- Elastic Net regularization allows for a balance between methods to determine when you want features to go to 0 or not

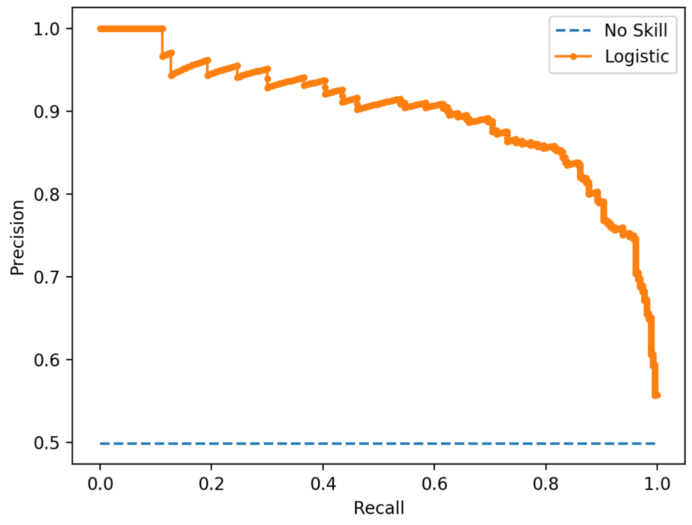
- L1 regularization will encourage most of the non-informative weights to be exactly 0.

3. (20 points) Calculations

For the following parts of this question, recall that

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives}$$
$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives}$$

Like an ROC curve (which uses False Positive Rate on the x axis and true positive rate on the y axis), a method by which we evaluate model performance is by the area under the curve (AUC) of the precision recall curve (PRC), illustrated here:

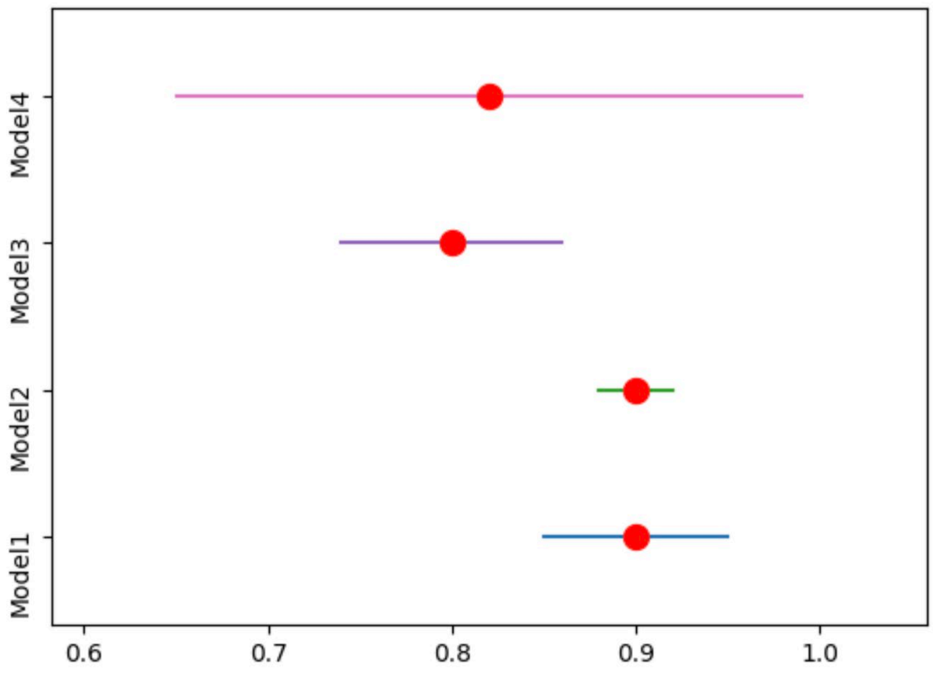


a) (10 points) Please calculate the precision and recall with thresholds p=0.45 and p=0.6. (You can approximate the fraction values). Use these values to draw a Precision and Recall Curve

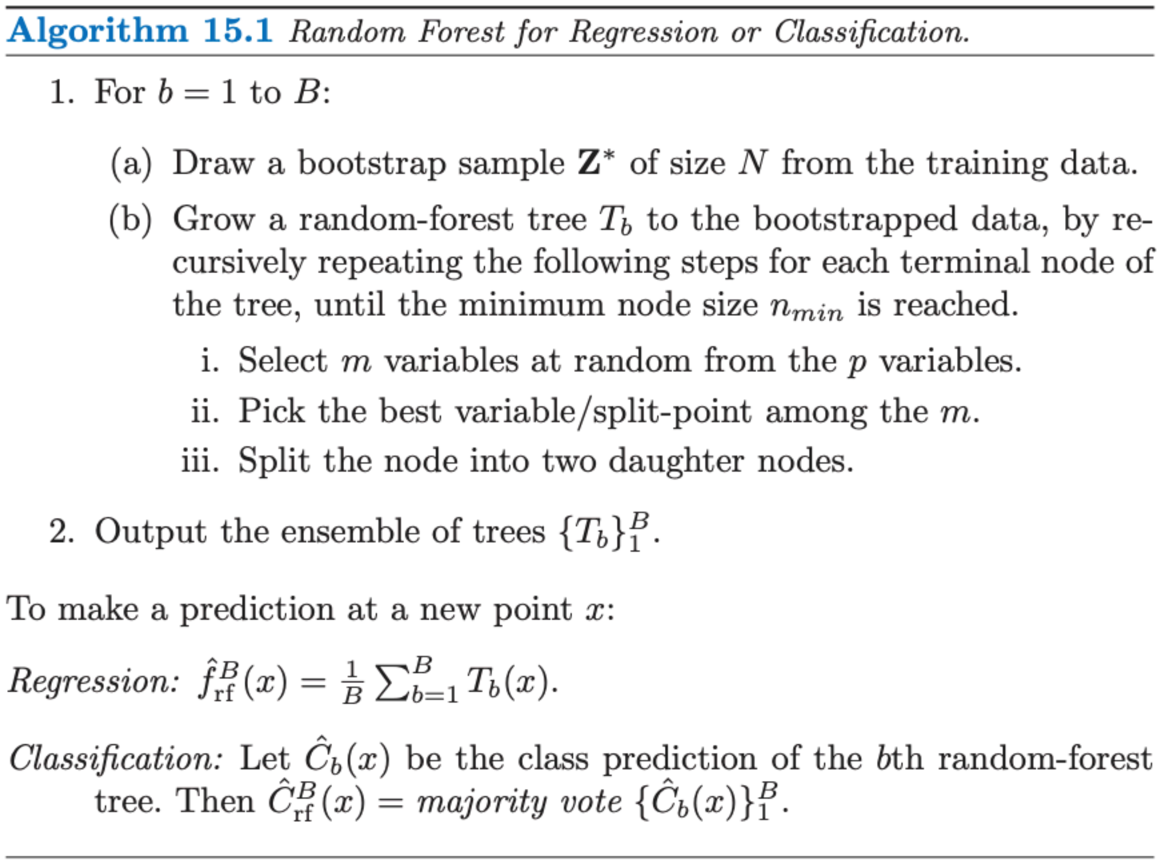
Predicted Probability	0	5	10	14	24	50	53	57	75	100
Ground Truth	No	No	No	Yes	No	Yes	Yes	No	Yes	Yes

This Page Intentionally Left Blank

b) (10 points) The confidence intervals for 4 different models' score are shown below. Compare the expected result of these models based on the plot. Which model do you expect to perform the best?



4. (30 Points) Debugging. Assume you have the following implementation of Random Forest, the algorithm of which is provided for you here:



Algorithm 15.1 *Random Forest for Regression or Classification.*

```
1. For $b = 1$ to $B$:
    (a) Draw a bootstrap sample $\mathbf{Z}^*$ of size $N$ from the training data.
    (b) Grow a random-forest tree $T_b$ to the bootstrapped data, by re-
        cursively repeating the following steps for each terminal node of
        the tree, until the minimum node size $n_{min}$ is reached.
        i. Select $m$ variables at random from the $p$ variables.
        ii. Pick the best variable/split-point among the $m$.
        iii. Split the node into two daughter nodes.
2. Output the ensemble of trees $\{T_b\}_1^B$.
```

To make a prediction at a new point x:

Regression: $\hat{f}_{\rm rf}^B(x) = \frac{1}{B}\sum_{b=1}^B T_b(x)$.

Classification: Let $\hat{C}_b(x)$ be the class prediction of the bth random-forest tree. Then $\hat{C}_{\rm rf}^B(x) = \textit{majority vote}\ \{\hat{C}_b(x)\}_1^B$.

a) We defined two functions, F1 and F2 as below which are being used in the function F3.

```
1)  def F1(p):
2)      if p == 0:
3)          return 0
4)      elif p == 1:
5)          return 0
6)      else:
7)          return - (p * np.log2(p) + (1 - p) * np.log2(1-p))
8)
9)  def F2(left_child, right_child):
10)     parent = left_child + right_child
11)     p_parent = parent.count(1) / len(parent) if len(parent) > 0 else 0
12)     p_left = left_child.count(1) / len(left_child) if len(left_child) > 0 else 0
13)     p_right = right_child.count(1) / len(right_child) if len(right_child) > 0 else 0
14)     metric_p = F1(p_parent)
15)     metric_l = F1(p_left)
16)     metric_r = F1(p_right)
17)     return metric_p - len(left_child) / len(parent) * metric_l - len(right_child) /
    len(parent) * metric_r
```

The function F2 is used in the function F3 as shown below (line 10.g):

```
1) def F3(X_bootstrap, y_bootstrap, max_features):
2) feature_ls = list()
3) num_features = len(X_bootstrap[0])

4) while len(feature_ls) <= max_features:
5) feature_idx = random.sample(range(num_features), 1)
6) if feature_idx not in feature_ls:
     a. feature_ls.extend(feature_idx)

7) metric = -999
8) node = None
9) for feature_idx in feature_ls:
10) for split_point in X_bootstrap[:,feature_idx]:
     a. left_child = {'X_bootstrap': [], 'y_bootstrap': []}
     b. right_child = {'X_bootstrap': [], 'y_bootstrap': []}

        # continuous variables
```

```
    c. if type(split_point) in [int, float]:
          i.  for i, value in enumerate(X_bootstrap[:,feature_idx]):
                 1. if value <= split_point:
                 2. left_child['X_bootstrap'].append(X_bootstrap[i])
                 3. left_child['y_bootstrap'].append(y_bootstrap[i])
                 4. else:
                 5. right_child['X_bootstrap'].append(X_bootstrap[i])
                 6. right_child['y_bootstrap'].append(y_bootstrap[i])
       # categorical variables
    d. else:
          i.  for i, value in enumerate(X_bootstrap[:,feature_idx]):
                 1. if value == split_point:
                       a. left_child['X_bootstrap'].append(X_bootstrap[i])
                       b. left_child['y_bootstrap'].append(y_bootstrap[i])
                 2. else:
                       a. right_child['X_bootstrap'].append(X_bootstrap[i])
                       b. right_child['y_bootstrap'].append(y_bootstrap[i])

    e. split_metric = F2(left_child['y_bootstrap'], right_child['y_bootstrap'])
    f. if split_metric > metric:
          i.  metric = split_metric
         ii.  left_child['X_bootstrap'] = np.array(left_child['X_bootstrap'])
        iii.  right_child['X_bootstrap'] = np.array(right_child['X_bootstrap'])
         iv.  node = {'Metric': split_metric,
                 1. 'left_child': left_child,
                 2. 'right_child': right_child,
                 3. 'split_point': split_point,
                 4. 'feature_idx': feature_idx}
11) return node
```

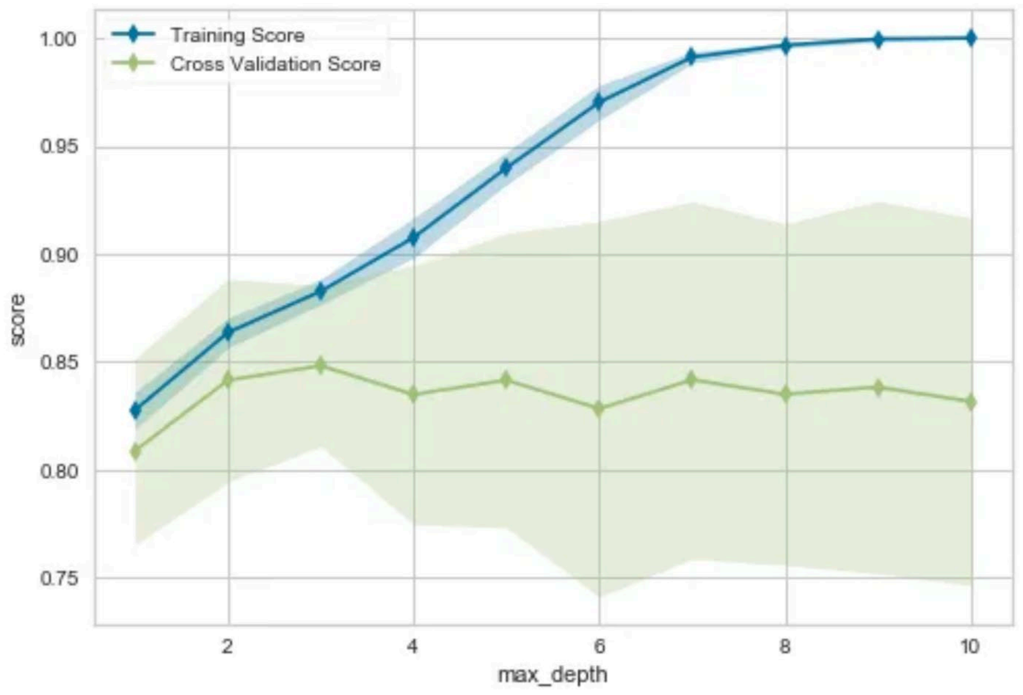
Please write the actual names of F1 and F2 as well as the mathematical formulation for these two functions. Furthermore, explain which line(s) of F3 correspond to parts b.i, b.ii and b.iii of the provided random forest algorithm:

i. Select m variables at random from the p variables.
ii. Pick the best variable/split-point among the m.
iii. Split the node into two daughter nodes.

This Page Intentionally Left Blank

When testing this model, it does not seem to work correctly.

b) In particular we found the following learning curve when changing the max_depth parameter (check function F4).



```
def F4(node, max_features, min_samples_split, max_depth, depth):
    left_child = node['left_child']
    right_child = node['right_child']

    del(node['left_child'])
    del(node['right_child'])

    if len(left_child['y_bootstrap']) == 0 or len(right_child['y_bootstrap']) == 0:
        empty_child = {'y_bootstrap': left_child['y_bootstrap'] + right_child['y_bootstrap']}
        node['left_split'] = terminal_node(empty_child)
        node['right_split'] = terminal_node(empty_child)
        return

    if depth >= max_depth:
        node['left_split'] = terminal_node(left_child)
        node['right_split'] = terminal_node(right_child)
        return node

    if len(left_child['X_bootstrap']) <= min_samples_split:
        node['left_split'] = node['right_split'] = terminal_node(left_child)
    else:
        node['left_split'] = F3(left_child['X_bootstrap'], left_child['y_bootstrap'],
max_features)
        F4(node['left_split'], max_depth, min_samples_split, max_depth, depth + 1)
    if len(right_child['X_bootstrap']) <= min_samples_split:
        node['right_split'] = node['left_split'] = terminal_node(right_child)
    else:
        node['right_split'] = F3(right_child['X_bootstrap'], right_child['y_bootstrap'],
max_features)
        F4(node['right_split'], max_features, min_samples_split, max_depth, depth + 1)
```

Explain what is happening in this case and the reason(s) behind it. Additionally describe how you can mitigate the issue happening here.

This Page Intentionally Left Blank