Spatial Data Mining: Homework 4 (Due 11/28 at 11:59PM)

Use blue color to write your answers and submit on ELMS.

Writing Problems

1. (15 points) Concepts:

(1) Select all true statements about training, validation and test data:
   A. Both training and validation data are used in computing gradients (e.g., linear regression) during training
   B. Only training data are used in computing gradients True
   C. Both validation and test data can be used to fine-tune a machine learning model (e.g., help select learning rates)
   D. Only validation data can be used to fine-tune a machine learning model True

(2) A machine learning method's performance in real applications is best reflected by its performance on:
   A. Training data
   B. Validation data
   C. Test data True
   D. Average of the three

(3) Which of the following are related to overfitting?
   A. High accuracy on test data but poor on training data
   B. High accuracy on test data but poor on validation data
   C. High accuracy on training data but poor on test data True
   D. High accuracy on training data but poor on validation data True
   E. Using complex models (e.g., deep decision tree, high-order polynomials) for simple phenomenonTrue
   F. Using simple models (e.g., shallow decision tree, linear regression) for complex phenomenon

(4) Select all true statements about logistic regression:
   A. It is used to predict discrete class labels (i.e., classification) True
   B. It is used to predict continuous values (i.e., regression)
   C. It assumes output labels are linear in input features (i.e., an affine transformation of input features)
   D. It assumes log-odds ratios are linear in input features True
   E. In order to calculate the likelihood, it requires assumptions on the distribution of input features (i.e., $p(\boldsymbol{x})$) due to the use of Bayes' rule; for example, by assuming $p(\boldsymbol{x})$ follows a normal distribution

(5) Select all true statements about decision tree and random forest:
   A. Each tree in a random forest is often weaker (less accurate) than a typical decision tree (that is not used as a part of a random forest) True
   B. Random forest prefers its trees to be strongly correlated
   C. Random forest prefers its trees to be uncorrelated True
   D. Two trees in a random forest are identical if their bootstrapped data are the same True
   E. A decision tree can often outperform a random forest on training data (assuming both are trained optimally for training data)

2. (25 points) Calculation: Decision Tree and Random Forest.

The following restaurant table used in class contains 12 data samples (X_1 to X_{12}), where each sample has a label (last column) showing whether the customer will wait at the restaurant, as well as 10 features/attributes that describe the situation. In the table, "T" means true and "F" means false.

Example	*Attributes*										*Target*
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	WillWait
X_1	T	F	F	T	Some	$$$	F	T	French	0–10	T
X_2	T	F	F	T	Full	$	F	F	Thai	30–60	F
X_3	F	T	F	F	Some	$	F	F	Burger	0–10	T
X_4	T	F	T	T	Full	$	F	F	Thai	10–30	T
X_5	T	F	T	F	Full	$$$	F	T	French	>60	F
X_6	F	T	F	T	Some	$$	T	T	Italian	0–10	T
X_7	F	T	F	F	None	$	T	F	Burger	0–10	F
X_8	F	F	F	T	Some	$$	T	T	Thai	0–10	T
X_9	F	T	T	F	Full	$	T	F	Burger	>60	F
X_{10}	T	T	T	T	Full	$$$	F	T	Italian	10–30	F
X_{11}	F	F	F	F	None	$	F	F	Thai	0–10	F
X_{12}	T	T	T	T	Full	$	F	F	Burger	30–60	T

Recall the two key steps that introduces randomness to each tree in a random forest: bootstrapping and random features. Suppose in a tree, the 12 **bootstrapped data samples** are:

[X_1, X_4, X_1, X_9, X_{10}, X_6, X_3, X_{11}, X_{12}, X_3, X_8, X_{10}]

And suppose at each split we consider two **random features**, and for the first split (root level), the two randomly sampled features are "Fri" (Friday) and "Est" (Estimated wait time).

Select the best feature (better of the two) to use for this split by comparing the information gain values: **Fill out the following tables, and show intermediate steps after the tables if you are unsure about whether your calculation is correct.**

"Fri":

Branch	**Number of samples with Label = T**	**Number of samples with Label = F**
"Fri" = T	2	3
"Fri" = F	6	1

"Est":

Branch	**Number of samples with Label = T**	**Number of samples with Label = F**
"Est" in 0-10	6	1
"Est" in 10-30	1	2
"Est" in 30-60	1	0
"Est" > 60	0	1

Calculation:

Quantity	**Branch: "Fri"**	**Branch: "Est"**
H(X)	0.92	0.92
H(X|Y)	0.75	`0.57
IG = H(X) – H(X|Y)	0.17	0.35

3. (60 points) Programming.

3.1 Linear regression (30 points). **USE HW4.ipynb (upload it into your Google CoLab).**

Given input features $\boldsymbol{X} \in R^{n\times d}$, and labels $\boldsymbol{y} \in R^{n\times 1}$. Linear regression models predicts $\hat{\boldsymbol{y}}$ as $\boldsymbol{Xw}$, where the optimal solution for $\boldsymbol{w} \in R^{d\times 1}$ is $\boldsymbol{w} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}$ (if mean squared loss is used).

(1) For the following data where n=10 and d = 1, put it as a NumPy array in CoLab notebook and implement the above solution (cannot use LinearRegression() function from scikit-learn and alike) to find the optimal value of $\boldsymbol{w}$ (as d=1, the solution only contains one number), based on the demo from class.

Copy your **CODE** here and report the optimal $\boldsymbol{w}$**.**

Hint: Pay attention to the dimensions when implementing it yourself.

$\boldsymbol{X}^T = [1,4,6,2,5,3,14,100,20,39]$ $\boldsymbol{y}^T = [5,16,20,10,21,14,63,449,88,172]$

```
from numpy.linalg import inv

X = np.asarray([1,4,6,2,5,3,14,100,20,39]).reshape([-1,1])

y = np.asarray([5,16,20,10,21,14,63,449,88,172]).reshape([-1,1])

def LR_solver(X,y):

  w = np.dot(inv(np.matmul(X.T, X)), np.matmul(X.T, y))

        return w

w = 4.47296855
```

(2) Change 449 in $\boldsymbol{y}$ (8th sample) to 1000 and calculate optimal $\boldsymbol{w}$ again. Copy your $\boldsymbol{w}$ solution here. Is linear regression sensitive to outliers in data?

w = 8.98640236

Yes, it is sensitive to outliers in the data.

(3) Try your code on the **diabetes** data shown in class. Replace X and y Use 40% of data as test data (data code provided in HW4.ipynb). Copy your $\boldsymbol{w}$ solution here.

```
[406.21780773 -658.34752664 537.99291519 516.11138505 1018.76961307
 -536.86438588 -1304.00932508 -787.4022709
   -62.21302281 -152.56453142]
```

(4) Verify your result using the linear regression function from scikit-learn. Note that you should set the "fit_intercept" property to "False", i.e., the call should be: myLR = LinearRegression(fit_intercept=False) Copy your $\boldsymbol{w}$ solution from scikit-learn here and check if it is the same as your code's result.

```
[406.21780773 -658.34752664 537.99291519 516.11138505 1018.76961307
 -536.86438588 -1304.00932508 -787.4022709
   -62.21302281 -152.56453142]
```

3.2 (30 points) Decision tree and random forest.

Load the **digits data (NOT the diabetes data from the last question)** from scikit-learn as demo-ed in class, and use 40% as test data.

(1) Train the decision tree classifier from scikit-learn (included in HW4.ipynb) and fill the following table:

Decision tree parameter	Accuracy on training data	Accuracy on test data
max_depth = 5	0.8608534322820037	0.8052851182197497
max_depth = 10	1.0	0.8720445062586927

(2) Train the random forest classifier from scikit-learn (included in HW4.ipynb) and fill the following table:

Random forest parameter	Accuracy on training data	Accuracy on test data
max_depth = 5	0.9777365491651205	0.9415855354659249
max_depth = 10	1.0	0.9694019471488178

*Set random_state=0 and max_features = 8 (included in code).

(3) Compare the two methods in terms of their overall performance on test data as well as their generalizability (i.e., how consistent are the accuracies on training and test data).

In General, Random Forest outperforms Decision Trees in terms of accuracies on both training and testing datasets. Decision Trees tends to overfit more on training data than Random Forest does especially when the tree is particularly deep as we can see in the case of max_depth of 10 – it got an accuracy of a %100 on the training data and performed less on the test set.

(4) Set max_features = 64 and max_depth = 10 for random forest and get the accuracies on training and test data. What's the impact by increasing the number of candidate features? Briefly explain why (hint: for this data, 64 means all features are used).

Accuracy on training data: 0.99

Accuracy on testing data: 0.94

Increasing the number of candidate features increases the training accuracy, and overfitting occurs because the performance drops on the test dataset.