Midterm Exam - Applied Machine Learning COMS W4995

Date: 03/08/17

Name:

UNI:

1 True/False (+/- 2pt each)

	True	False
A git rebase will change a commit's hash.	X	
A larger number of neighbors in KNeighborsClassifier corresponds to a more complex model.		X
Complex models tend to overfit more.	X	
Trying different models on the same test set leads to overly optimistic evaluation results.	X	
Linear Regression works well when the data has more feature than samples.		X
Increasing L1 penalty in Logistic Regression leads to less non-zero coefficients.	X	
Regression models based on minimizing mean squared error are sensitive to outliers.	X	
One-vs-one classification builds n_classes^2 / 2 classifiers.		X
A decision tree without pruning will always achieve 100% training set accuracy.	X	
A better calibrated classifier is always more accurate.		X

2 Multiple choice (5pt / question)

Select all choices that apply.

2.1 What are the benefits of cross-validation compared to using a single split of the data?

- ✓ Potentially more data to build the model during evaluation.
- ❏ Faster.
- ✓ Provides uncertainty estimate of the evaluation result.
- ✓ Each datapoint is used as test point.

Points: 5/3/2/1

2.2 When should you eliminate a feature?

- ❏ When is has low variance.
- ❏ When it only has two different values on the whole training set.
- ✓ When it has the same value for nearly all samples.

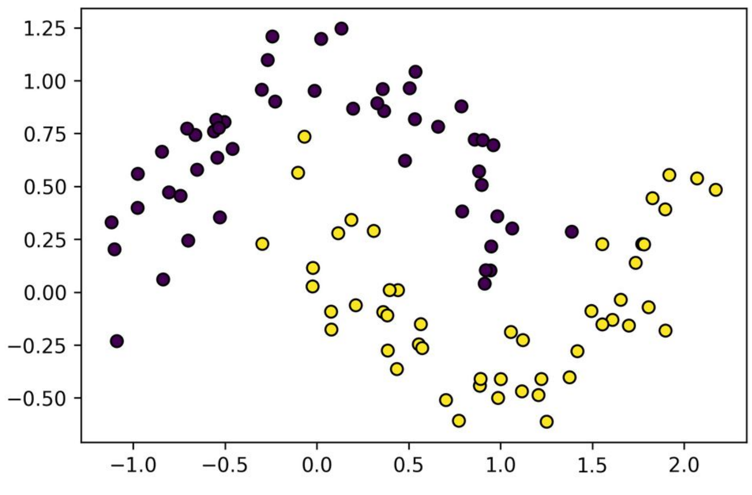
Points: 5/3/1/0

2.3 Which of the following models are sensitive to the scaling of the data?

- ❏ Decision Trees
- ❏ Random Forests
- ✓ Kernel SVM
- ✓ Logistic Regression
- ✓ Nearest Neighbors
- ❏ Gradient Boosted Trees

Points: 5/4/3/3/2/1/0

2.4 Which of the following models can achieve zero training error on the two-class classification dataset shown below?



- ❏ LogisticRegression
- ✓ Decision Tree
- ✓ RBF-Kernel SVM
- ✓ Linear SVM with Polynomial features
- ❏ Nearest Shrunken Centroid

Points: 5/4/3/2/1/0

3 Debugging (10pt each)

For each code snippet, find and explain all errors given the task. Assume all necessary imports have been made. There can be more than one error per task!

3.1 Task: Scale the data present in X_train and X_test, build a ridge regression model on the scaled training set and evaluate the model on the scaled test set.

```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
ridge = Ridge(alpha=1).fit(X_train, y_train)
X_test_scaled = scaler.fit_transform(X_test)
score = ridge.score(X_test_scaled, y_test)
```

Ridge is fit on unscaled data. (3pts)
Scaler shouldn’t be fit again on test data. (8pts)

3.2 Task: Apply logistic regression to a dataset consisting only of categorical variables, and having missing values and visualize the 10 most important coefficients. Assume that feature_names is a list of length n_features containing strings describing the features.

```
pipe = make_pipeline(Imputer(strategy=”mean”), OneHotEncoder(),
                                    LogisticRegression())
pipe.fit(X_train, y_train)
coef = pipe.coef_
important = np.argsort(coef)[-10:]
plt.barh(range(10), coef[important])
plt.xticks(range(10), feature_names[important])
```

Mean imputation is meaningless for categorical data (5pts)
Pipe doesn’t have a “coef_” attribute (5pts)
Argsort should use abs for important features. (5pts)
Feature names don’t correspond to one-hot encoded features. (5pts)

4 Coding (10 each)

Assume all necessary imports have been made.

4.1 Provide code to implement grid-searching the maximum depth of a DecisionTreeClassifier between 1 and 10 (inclusive 10), using 10-fold cross-validation. Evaluate the best parameter setting on a separate test set, given the whole dataset as numpy arrays X and y.

```
param_grid = {'max_depth': range(1, 11)}   3pt
X_train, X_test, y_train, y_test = train_test_split(X, y, stratify=y) 1 pt
grid = GridSearchCV(DecisionTreeClassifier(), param_grid, cv=10) 4pt
grid.fit(X_train, y_train) 1pt
score = grid.score(X_test, y_test) 1pt
```

4.2 Write code that is equivalent to the following without using a pipeline:

```
pipe = make_pipeline(StandardScaler(), SelectPercentile(),
Ridge())
pipe.fit(X_train, y_train)
score = pipe.score(X_test, y_test)
```

```
scaler = StandardScaler()                  1 instantiating
percentile = SelectPercentile()      1 instantiating
X_train_scaled = scaler.fit_transform(X_train)  1 fit, 1 transform
X_train_selected = percentile.fit_transform(X_train_scaled,
                                                  y_train)  1fit, 1
transform
ridge = Ridge().fit(X_train_selected, y_train) 1 fit
X_test_selected = precentile.transform(scaler.transform(X_test)) 1 each
score = ridge.score(X_test_selected, y_test) 1 score
```

-2 if refitting on test. No points for transform without storing output.

5 Concepts (5pt each)

Answer each question with a short (2-5 sentences) explanation.

5.1 What is the difference between quantitative, sequential and diverging colormaps?
Quantitative: for categorical variables, markers, lines. Discrete colors with strong contrast between the colors.
Sequential: Changing from one color continuously to another, usually from one hue to another and one lightness to another. Used for continuous variables.
Diverging: Has a defined mid-point and goes to two different colors towards the end-points. Used for continuous variables with a defined zero point.

5.2 Explain the difference between grid-search and cross-validation.
Grid-search is a strategy to find good hyper-parameter settings by brute-force search, that is by trying all combinations under consideration.
Cross-validation is a way to robustly estimate the generalization performance of one specific model (with one specific parameter setting).

5.3 Explain how the one-vs-rest multi-class classification method is used for training and prediction.
OvR is a strategy for reducing a multi-class classification problem to a series of binary classification problems, so that a binary classifier can be used on multi-class problems.
One classifier is trained classifying each class against all the other classes.
To make a prediction, all classifiers are applied and the class with the most certain outcome (highest probability or decision function) is chosen as prediction.

5.4 Explain k-nearest-neighbors imputation.
A strategy to fill in missing data. For each data point with missing values, the k (some integer number, e.g. 5) closest points are computed, using the features that are not missing. The missing values are then set to the (optionally distance-weighted) mean of these features among the neighbors. Bonus: distances is computed using only features available in the neighbor and the query point, and divided by the number of features (mean distance).