Economics, Problem Set #7, Machine Learning

OSE Lab, Dr. Evans
Due Friday, Aug. 9 at 11:00pm

1. **Multinomial logistic regression and cross validation.** For this problem, you will estimate the probability that a given wine comes from a given *cultivar*. The data in the file strongdrink.txt (taken from the UCI Machine Learning Repository) are the results of a chemical analysis of 176 Italian wines from three known cultivars (a cultivar is a group of grapes selected for desirable characteristics that can be maintained by propagation). The chemical analysis determined the quantities of the following 13 different constituents (the last 13 variables):

Variable	Name	Variable	Name
Alcohol	`alco`	Nonflavanoid phenols	`nonfl_phen`
Malic acid	`malic`	Proanthocyanins	`proanth`
Ash	`ash`	Color intensity	`color_int`
Alkalinity of ash	`alk`	Hue	`hue`
Magnesium	`magn`	OD280/OD315 of diluted wines	`OD280rat`
Total phenols	`tot_phen`	Proline	`proline`
Flavanoids	`flav`		

   (a) Use a multinomial logistic regression model of the following form with the following linear predictor η_j for $j = 1, 2$ (the baseline class is $j = 3$).

$$Pr(cultivar_i = j|X\beta_j) = \frac{e^{\eta_j}}{1 + \sum_{j=1}^{J-1} e^{\eta_j}} \quad \text{for} \quad j = 1, 2$$

$$\text{where} \quad \eta_j = \beta_{j,0} + \beta_{j,1} alco_i + \beta_{j,2} malic_i + \beta_{j,3} tot\_phen_i + \beta_{j,4} color\_int_i$$

   Estimate the model on a 75% sample training set using the following command. Report your two sets of estimated coefficients and intercepts for $j = 1$ and $j = 2$ (not the coefficients for $j = 3$). Report your error rates (1 - precision) on the test set using the code below. Which category(ies) of cultivar is the model best at predicting? Is (are) the most accurately predicted category(ies) the one(s) with the most observations? Report the MSE from the test set.

```
from sklearn.cross_validation import train_test_split
from sklearn.metrics import classification_report

X_train, X_test, y_train, y_test = \
    train_test_split(X, y, test_size = 0.25,
        random_state=20)
print(classification_report(y_test, y_pred))
```

(b) Perform a leave-one-out cross validation (LOOCV) with the model from part (a). Report your error rates (1 - precision) for each category? How do your error rates compare to those from part (a)? Report your LOOCV estimate for the test MSE as the average MSE, where y_i is the left out observation from each test set.

$$CV_{loo} = \frac{1}{N}\sum_{i=1}^{N} MSE_i = \frac{1}{N}\sum_{i=1}^{N}\Big[1 - I\left(y_i = \hat{y}_i\right)\Big]$$

(c) Perform a k-fold cross validation in which the data are divided into $k = 4$ groups. Use the following code. Report your error rates (1 - precision) for each category. How do your error rates compare to those from parts (a) and (b)? Report your k-fold estimate for the test MSE as the average MSE.

```
from sklearn.model_selection import KFold

kf = KFold(n_splits=4, shuffle=True, random_state=10)
kf.get_n_splits(X)
```

$$CV_{kf} = \frac{1}{k}\sum_{i=1}^{k} MSE_i \quad \text{where} \quad MSE_i = \frac{1}{n}\sum_{j=1}^{N}\Big[1 - I\left(y_j = \hat{y}_j\right)\Big]$$

2. **Decision trees.**[1] Joe Biden was the 47th Vice President of the United States. He was the subject of many memes, attracted the attention of Leslie Knope (Parks and Recreation, TV sitcom), and experienced a brief surge in attention due to photos from his youth. The data file `biden.csv` contains a selection of variables from the 2008 American National Election Studies survey that allow you to test competing factors that may influence attitudes towards Joe Biden. The variables are coded as follows:

- `biden`: feeling thermometer ranging from 0 to 100. Feeling thermometers are a common metric in survey research used to gauge attitudes or feelings of warmth towards individuals and institutions. They range from 0-100, with 0 indicating extreme coldness and 100 indicating extreme warmth.
- `female`: =1 if respondent is female, =0 if respondent is male
- `age`: age of respondent in years, range from 18 to 93
- `dem`: =1 if respondent is a Democrat, =0 otherwise
- `rep`: =1 if respondent is a Republican, =0 otherwise
- `educ`: number of years of formal education completed by respondent, range from 0 to 17 with 17+ representing the first year of grad school and up.

[1]This problem was originally created by Benjamin Soltoff as an application for the R programming language.

(a) Split the data into a training set (70%) and a test set (30%) using the `sklearn.model_selection.train_test_split()` function with `random_state=25`. Setting the seed will guarantee you all get the same results. Use recursive binary splitting to fit a decision tree to the training data, with `biden` as the response variable and the other variables as predictors. Set the `max_depth=3` and `min_samples_leaf=5` Plot the tree and interpret the results. What is the test MSE?

(b) Use `sklearn.model_selection.RandomizedSearchCV` to optimally tune the hyperparameters in the decision tree from part (a). Tune the parameters `max_depth`, `min_samples_split`, and `min_samples_leaf`. Set `n_iter=100`, `n_jobs=-1`, `cv=5` for $k = 5$ k-fold cross validation, `random_state=25`, and `scoring='neg_mean_squared_error'`. This last option will allow you to compare the MSE of the optimized tree (it will output the negative MSE) to the MSE calculated in part (a). Set your parameter distributions over which to test random combinations to the following.

```
     from scipy.stats import randint as sp_randint

     param_dist1 = {'max_depth': [3, 10],
                    'min_samples_split': sp_randint(2, 20),
                    'min_samples_leaf': sp_randint(2, 20)}
```

Report your optimal tuning parameter values (use the `.best_params_` object of your `RandomizedSearchCV().fit(X, y))` results). Report the MSE of your optimal results (use the `.best_score_` object of your `RandomizedSearchCV().fit(X, y))` results.

(c) Now tune the parameters of a RandomForest regression model on these data `sklearn.ensemble.RandomForestRegressor()`. Use `sklearn.model_selection.RandomizedSearchCV` to optimally tune the hyperparameters in the random forest regression model. Tune the parameters `n_estimators`, `max_depth`, `min_samples_split`, `min_samples_leaf`, and `max_features`. Set `n_iter=100`, `n_jobs=-1`, `cv=5` for $k = 5$ k-fold cross validation, `random_state=25`, and `scoring='neg_mean_squared_error'`. Set your Random Forest parameter distributions over which to test random combinations to the following.

```
     param_dist2 = {'n_estimators': [10, 200],
                    'max_depth': [3, 10],
                    'min_samples_split': sp_randint(2, 20),
                    'min_samples_leaf': sp_randint(2, 20),
                    'max_features': sp_randint(1, 5)}
```

Report your optimal tuning parameter values (use the `.best_params_` object of your `RandomizedSearchCV().fit(X, y))` results). Report the MSE of your optimal results (use the `.best_score_` object of your `RandomizedSearchCV().fit(X, y))` results.

3. **Classifier "horse" race.** For this problem, you will use the 397 observations from the Auto.csv dataset.[2] This dataset includes 397 observations on miles per gallon (`mpg`), number of cylinders (`cylinders`), engine displacement (`displacement`), horsepower (`horsepower`), vehicle weight (`weight`), acceleration (`acceleration`), vehicle year (`year`), vehicle origin (`origin`), and vehicle name (`name`). We will study the factors that make miles per gallon high or low. Create a binary variable `mpg_high` that equals 1 if `mpg`$\geq$ median(`mpg`) and equals either 0 if `mpg`$<$ median(`mpg`).

   (a) Use `sklearn.linear_model.LogisticRegression` to fit a logistic model of `mpg_high` on features number of cylinders (`cyl`), engine displacement (`dspl`), horsepower (`hpwr`), vehicle weight (`wgt`), acceleration (`accl`), vehicle year (`yr`), vehicle origin 1 (`orgn1`), and vehicle origin 2 (`orgn2`). Make sure to include a constant term. Fit the model using k-fold cross validation with $k = 4$ folds.[3]

   ```
       kf_log = KFold(n_splits=4, shuffle=True, random_state=25)
   ```

   Report the MSE of the model as the average MSE across the $k = 4$ test sets, and report the error rates for each category of `mpg_high` as the average error rate for that category across the $k = 4$ test sets.

   $$Pr(mpg\_high = 1|\boldsymbol{X\beta}) = \frac{e^{\boldsymbol{X\beta}}}{1 + e^{\boldsymbol{X\beta}}}$$
   $$\text{where} \quad \boldsymbol{X\beta} = \beta_0 + \beta_1 cyl_i + \beta_2 dspl_i + \beta_3 hpwr_i + \beta_4 wgt_i + \beta_5 accl_i + \beta_6 yr_i + \beta_7 orgn1_i + \beta_8 orgn2_i$$

   The variables $orgn1_i$ and $orgn2_i$ are indicator variables for two of the three categories of the $orgn_i$ variable (values 1 and 2, with value 3 left out). Also, you will need to drop the $name_i$ variable, which is a string variable that gives the name of the car.

   (b) Use `sklearn.ensemble.RandomForestClassifier` to fit a random forest model of `mpg_high` on the eight possible features used in part (a). Use `sklearn.model_selection.RandomizedSearchCV` to optimally tune the hyperparameters in the random forest classification model. Tune the parameters `n_estimators`, `max_depth`, `min_samples_split`, `min_samples_leaf`, and `max_features`. Set `n_iter=100`, `n_jobs=-1`, `cv=4` for $k = 4$ k-fold cross validation, `random_state=25`, and `scoring='neg_mean_squared_error'`. Set your Random Forest parameter distributions over which to test random combinations to the following.

   ```
       param_dist3 = {'n_estimators': [10, 200],
                      'max_depth': [3, 8],
   ```

[2]The Auto.csv dataset comes from James et al. (2017, Ch. 3) and is available at http://www-bcf.usc.edu/ gareth/ISL/data.html.

[3]`sklearn.model_selection.KFold`.

```
                         'min_samples_split': sp_randint(2, 20),
                         'min_samples_leaf': sp_randint(2, 20),
                         'max_features': sp_randint(1, 8)}
```

Report your optimal tuning parameter values (use the `.best_params_` object of your `RandomizedSearchCV().fit(X, y))` results). Report the MSE of your optimal results (use the `.best_score_` object of your `RandomizedSearchCV().fit(X, y))` results.

(c) Use `sklearn.svm.SVC` to fit a support vector machines model of `mpg_high` with a Gaussian radial basis function kernel `kernel='rbf'` on the eight features used in parts (a) and (b). Use `sklearn.model_selection.RandomizedSearchCV` to optimally tune the hyperparameters in the support vector machines classifier model. Tune the parameters `C` penalty parameter, `gamma` kernel coefficient, and `shrinking`. Set `n_iter=100`, `n_jobs=-1`, `cv=4` for $k = 4$ k-fold cross validation, `random_state=25`, and `scoring='neg_mean_squared_error'`. Set your SVM parameter distributions over which to test random combinations to the following.

```
    from scipy.stats import uniform as sp_uniform

    param_dist4 = {'C': sp_uniform(loc=0.2, scale=4.0),
                   'gamma': ['scale', 'auto'],
                   'shrinking': [True, False]}
```

Report your optimal tuning parameter values (use the `.best_params_` object of your `RandomizedSearchCV().fit(X, y))` results). Report the MSE of your optimal results (use the `.best_score_` object of your `RandomizedSearchCV().fit(X, y))` results.

(d) Which of the above three models do you think is the best predictor of `mpg_high`? Why?

4. **Neural network horse race.** For this problem, you will test the predictive accuracy of three models on classifying wines into one of three possible *cultivars*. The data in the file `strongdrink.txt` (the same data as in exercise 1, see table). The data are comprised of 176 observations, each of which is a chemical analysis of an Italian wine. Each wine is from one of three known cultivars (a cultivar is a group of grapes selected for desirable characteristics that can be maintained by propagation). The chemical analysis determined the quantities of the following 13 different constituents (the last 13 variables):

   (a) Create a scatterplot of the data where the x-variable is alcohol (*alco*) and the y-variable is color intensity (*color_int*). Make the dot of each of the three possible *cultivar* types a different color. Make sure your plot has a legend.

   (b) Use `sklearn.linear_model.LogisticRegression` to fit a multinomial logistic model of `cultivar` on features alcohol (*alco*), malic acid (*malic*), total phenols (*tot_phen*), and color intensity (*color_int*) with the following linear predictor.

$$Pr(cultivar_i = j|X\beta_j) = \frac{e^{\eta_j}}{1 + \sum_{j=1}^{J-1} e^{\eta_j}} \quad \text{for} \quad j = 1, 2$$

$$\text{where} \quad \eta_j = \beta_{j,0} + \beta_{j,1} alco_i + \beta_{j,2} malic_i + \beta_{j,3} tot\_phen_i + \beta_{j,4} color\_int_i$$

Use `sklearn.model_selection.RandomizedSearchCV` to optimally tune the hyperparameters `penalty` and `C` in the Logistic regression model. Set `n_iter=200`, `n_jobs=-1`, `cv=5` for $k = 5$ k-fold cross validation, `random_state=25`, and `scoring='neg_mean_squared_error'`. This last option will allow you to compare the MSE of the optimized multinomial logit model (it will output the negative MSE). Set your parameter distributions over which to test random combinations to the following.

```
from scipy.stats import uniform as sp_uniform

param_dist1 = {'penalty': ['l1', 'l2']
               'C': sp_uniform(0.1, 10.0)}
```

Report your optimal tuning parameter values (use the `.best_params_` object of your `RandomizedSearchCV().fit(X, y))` results). Report the MSE of your optimal results (use the `.best_score_` object of your `RandomizedSearchCV().fit(X, y))` results.

(c) Use `sklearn.ensemble.RandomForestClassifier` to fit a random forest model of `cultivar` on the same four features used in part (b). Use `sklearn.model_selection.RandomizedSearchCV` to optimally tune the hyperparameters in the random forest classification model. Tune the parameters `n_estimators`, `max_depth`, `min_samples_split`, `min_samples_leaf`, and `max_features`. Set `n_iter=200`, `n_jobs=-1`, `cv=5` for $k = 5$ k-fold cross validation, `random_state=25`, and `scoring='neg_mean_squared_error'`. Set your Random Forest parameter distributions over which to test random combinations to the following.

```
from scipy.stats import randint as sp_randint

param_dist2 = {'n_estimators': sp_randint(10, 200),
               'max_depth': sp_rand_int(2, 4),
               'min_samples_split': sp_randint(2, 20),
               'min_samples_leaf': sp_randint(2, 20),
               'max_features': sp_randint(1, 4)}
```

Report your optimal tuning parameter values (use the `.best_params_` object of your `RandomizedSearchCV().fit(X, y))` results). Report the MSE of your optimal results (use the `.best_score_` object of your `RandomizedSearchCV().fit(X, y))` results.

(d) Use `sklearn.svm.SVC` to fit a support vector machines classifier model of `cultivar` with a Gaussian radial basis function kernel `kernel='rbf'` on the four features used in parts (b) and (c). Use `sklearn.model_selection.RandomizedSearchCV` to optimally tune the hyperparameters in the support vector machines classifier model. Tune the parameters `C` penalty parameter, `gamma` kernel coefficient, and `shrinking`. Set `n_iter=200`, `n_jobs=-1`, `cv=5` for $k = 5$ k-fold cross validation, `random_state=25`, and `scoring='neg_mean_squared_error'`. Set your SVM parameter distributions over which to test random combinations to the following.

```
    from scipy.stats import uniform as sp_uniform

    param_dist3 = {'C': sp_uniform(loc=0.1, scale=10.0),
                   'gamma': ['scale', 'auto'],
                   'shrinking': [True, False]}
```

Report your optimal tuning parameter values (use the `.best_params_` object of your `RandomizedSearchCV().fit(X, y))` results). Report the MSE of your optimal results (use the `.best_score_` object of your `RandomizedSearchCV().fit(X, y))` results.

(e) Use `sklearn.neural_network.MLPClassifier` to fit a multiple hidden layer neural network (multiple layer perceptron) model of `cultivar`. Use `sklearn.model_selection.RandomizedSearchCV` to optimally tune the hyperparameters in the MLP classifier model. Tune the parameters `hidden_layer_sizes`, `activation`, and `alpha`. Set `n_iter=200`, `n_jobs=-1`, `cv=5` for $k = 5$ k-fold cross validation, `random_state=25`, and `scoring='neg_mean_squared_error'`. Set your MLP parameter distributions over which to test random combinations to the following.

```
    param_dist4 = {'hidden_layer_sizes': sp_randint(1, 100),
                   'activation': ['logistic', 'relu'],
                   'alpha': sp_uniform(0.1, 10.0)}
```

Report your optimal tuning parameter values (use the `.best_params_` object of your `RandomizedSearchCV().fit(X, y))` results). Report the MSE of your optimal results (use the `.best_score_` object of your `RandomizedSearchCV().fit(X, y))` results.

(f) Which of the above three models do you think is the best predictor of `cultivar`? Why?

References

James, Gareth, Daniela Witten, Trevor Hastie, and Robert Tibshirani, *An Introduction to Statistical Learning with Applications in R* Springer Texts in Statistics, Springer, 2017.