Problem Set #9

MACS 30150, Dr. Evans
Due Monday, Mar. 16 at 11:59pm

1. **Neural network horse race (10 points).** 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`. You used these data in PS7, exercise 1. 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):

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) 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