COMP4220: Machine Learning, Spring 2021, Assignment 5

Due: Wednsday, April 7, 11pm

Please submit one pdf file for all questions.

You can type your answer for the first two questions in the below cell of each question using "Markdown" option!

**When turning in assignments after the due date, please clearly specify the number of late hours used.

P1. What is the fundamental idea behind Support Vector Machines?

Support Vector Machines aim to have the largest margin between the decision boundary separating two classes and their training instances by fitting the widest street between them.

P2. If a Decision Tree is underfitting the training set, is it a good idea to try scaling the input features?

Since decision trees are not affected by whether the data is scaled, it would not be a good idea to scale the input features as it would not do anything.

P3.If you have trained five different models on the exact same training data, and they all achieve 95% precision, is there any chance that you can combine these models to get better results? If so, how? If not, why?

You can combine these models to get better results through combining them into a voting ensemble. This would work as a voting ensemble is more affective on models that differ from each other.

Programming assignments (SVM, Decision Tree, Random Forest)

In [1]:
```
# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

In [2]:
```
# Importing the dataset
dataset = pd.read_csv('Social_Network_Ads.csv')
X = dataset.iloc[:, [2, 3]].values
y = dataset.iloc[:, 4].values
```

P4. Programming assignment(SVM)

P4.1 Split the dataset into the Training set(%75) and Test set(%25)

use "train_test_split" from sklearn.model_selection.

Use these variables:(X_train, X_test, y_train, y_test)

In [5]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size =
0.75, test_size = 0.25, random_state = 0)
```

P4.2 Apply Feature Scaling method for X_train and X_test

Hint: use StandardScaler.fit_transform(X_train) for "X_train" and use StandardScaler.transform(X_test) for "X_test"

In [6]:
```
from sklearn.preprocessing import StandardScaler
StandardScaler = StandardScaler()
X_train = StandardScaler.fit_transform(X_train)
X_test = StandardScaler.transform(X_test)
```

P4.3 Train the SVM model on the Training set,

(random_state = 0)

In [12]:
```
from sklearn.svm import SVC
SVM_Model = SVC(kernel = 'linear', random_state = 0)
SVM_Model.fit(X_train, y_train)
```

Out[12]:
```
SVC(kernel='linear', random_state=0)
```

P4.4 Predict the Test set results and show the result

In [8]:
```
y_pred = SVM_Model.predict(X_test)
```

P4.5 Make the Confusion Matrix and show the result

In [11]:
```
from sklearn.metrics import confusion_matrix
Confusion_Matrix = confusion_matrix(y_test, y_pred)
print(Confusion_Matrix)
```

```
[[66  2]
 [ 8 24]]
```

P5. Programming assignment(Decision Tree)

P5.1 Train the Decision Tree Classification on the Training set,

(criterion = 'entropy', random_state = 0)

In [13]:
```
from sklearn.tree import DecisionTreeClassifier
Decison_Tree = DecisionTreeClassifier(criterion = 'entropy', random_sta
te = 0)
Decison_Tree.fit(X_train, y_train)
```

Out[13]:
```
DecisionTreeClassifier(criterion='entropy', random_state=0)
```

P5.2 Predict the Test set results

In [14]:
```
y_pred = Decison_Tree.predict(X_test)
```

P5.3 Make the Confusion Matrix and show the result

In [15]:
```
Confusion_Matrix = confusion_matrix(y_test, y_pred)
print(Confusion_Matrix)
```

```
[[62  6]
 [ 3 29]]
```

P6. Programming assignment(Random Forest)

P6.1 Train the Random Forest Classification on the Training set,

(criterion = 'entropy', random_state = 0)

In [16]:
```
from sklearn.ensemble import RandomForestClassifier
Random_Forest = RandomForestClassifier(criterion = 'entropy', random_st
ate = 0)
Random_Forest.fit(X_train, y_train)
```

Out[16]:
```
RandomForestClassifier(criterion='entropy', random_state=0)
```

P6.2 Predict the Test set results

In [17]:
```
y_pred = Random_Forest.predict(X_test)
```

P6.3 Make the Confusion Matrix and show the

result

In [18]:
```
Confusion_Matrix = confusion_matrix(y_test, y_pred)
print(Confusion_Matrix)
```

```
[[63  5]
 [ 4 28]]
```

Train and fine-tune a Decision Tree for the moons dataset by following these steps

P7.1. Use make_moons(n_samples=10000, noise=0.4) to generate a moons dataset.

In [19]:
```
from sklearn.datasets import make_moons
X, y = make_moons(n_samples = 10000, noise = 0.4)
```

P7.2. Use train_test_split() to split the dataset into a training set and a test set.

In [20]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y)
```

P7.3. Use grid search with cross-validation (with the help of the GridSearchCV class) to find good hyperparameter values for a DecisionTreeClassifier.

Hint: try various values for max_leaf_nodes.

Train it on the full training set using these hyperparameters, and measure your model's performance on the test set. You should get roughly 85% to 87% accuracy.

By default, GridSearchCV trains the best model found on the whole training set (you can change this by setting refit=False), so you don't need to do it again.

In [39]:
```
from sklearn.model_selection import GridSearchCV
Hyperparameters = {'max_leaf_nodes': list(range(10, 10000))}
Grid_Search = GridSearchCV(DecisionTreeClassifier(random_state = 0), Hy
perparameters, verbose = 1, n_jobs = -1, cv = 2)
Grid_Search.fit(X_train, y_train)
```

```
Fitting 2 folds for each of 9990 candidates, totalling 19980 fits
```

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 8 concurrent work
ers.
[Parallel(n_jobs=-1)]: Done  56 tasks      | elapsed:    0.0s
[Parallel(n_jobs=-1)]: Done 2160 tasks      | elapsed:    4.0s
[Parallel(n_jobs=-1)]: Done 6160 tasks      | elapsed:   12.6s
[Parallel(n_jobs=-1)]: Done 11760 tasks      | elapsed:   24.9s
[Parallel(n_jobs=-1)]: Done 18960 tasks      | elapsed:   41.3s
[Parallel(n_jobs=-1)]: Done 19980 out of 19980 | elapsed:   43.6s finis
hed
```

Out[39]:
```
GridSearchCV(cv=2, estimator=DecisionTreeClassifier(random_state=0), n_
jobs=-1,
             param_grid={'max_leaf_nodes': [10, 11, 12, 13, 14, 15, 16,
17, 18,
                                             19, 20, 21, 22, 23, 24, 25,
26, 27,
                                             28, 29, 30, 31, 32, 33, 34,
35, 36,
                                             37, 38, 39, ...]},
             verbose=1)
```

P7.4. Evaluate the model's accuracy and show the result

In [40]:
```
from sklearn.metrics import accuracy_score
y_pred = Grid_Search.predict(X_test)
Accuracy_Score = accuracy_score(y_test, y_pred)
print(Accuracy_Score)
```

```
0.852
```

In [ ]: