EECS461/ECE523
MACHINE LEARNING
Fall 2018

ASSIGNMENT 3

Due Date: *Wednesday, December 26th, 2018, 23:59*

Assignment Submission: *Turn in your assignment by the due date through LMS**.** Prepare a single Jupyter Notebook (.ipynb) with the answers to all questions.* ***Name the file as <your first name>_<your last name>_ assignment3.ipynb.*** *Make sure to* ***use the sample Jupyter Notebook file provided to you as template****.*

All work in questions must be your own; you must neither copy from nor provide assistance to anybody else*. If you need guidance for any question, talk to the instructor or TA in office hours. You can also reach to TA Hacer Tilbeç at hacertilbec@std.sehir.edu.tr*

Late Assignment Policy: You have a total of ***4 days of late assignment*** turn-in allowance throughout this semester. For a single assignment, you can use ***a maximum of 2 late-days.*** You decide which assignments you are going to use your 4 late-days. After assignment due date/time, each 24-hours period is counted as one late date (i.e., if you submit late 1 hour or 23 hours, you use 1 late-date). It is your responsibility to keep track of your late days. If you are late more than 2 days for any assignment or you exhausted your late days, you get 0 from the late assignment (No exceptions)

DATA SET

In this assignment, you are going to perform some basic image recognition tasks. You will use the same dataset that you used in assignment 2. This time, we will focus on only eyewear prediction.

X_train, y_train, X_test and y_test are provided for you in the template notebook.

DECISION TREE (5 points)

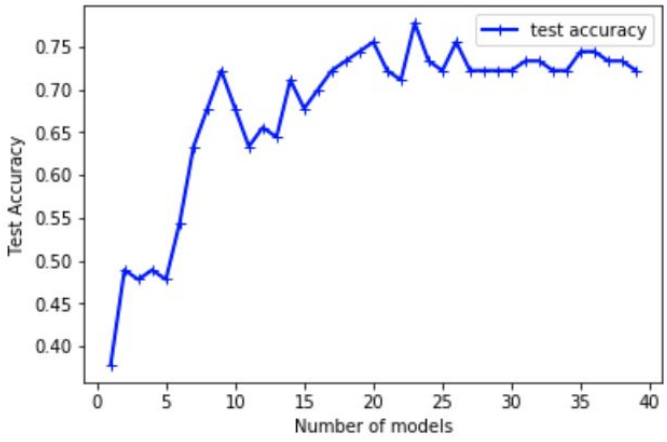
- **(a) (2 points)** Create a decision tree classifier with parameters of random_state=0 and max_depth=2. Report training and test accuracy of the model.
- **(b) (3 point)** Did your model performs well? Is bias low or high? What about variance?

BAGGING MODELS (60 points)

Bagging is a way to get a diverse set of models by using the same algorithm but training them on different training sets. As discussed in class, ***training instances can be sampled with replacement (bootstrap) or without replacement (also called pasting)***. ***Features can be sampled as well with or without replacement***. Sampling both training instances and features is referred to as Random Patches and sampling only features but keeping all training instances is referred to as Random Subspaces.

(c) (5 points) Create a bagging classifier with decision tree classifier (with parameters of random_state=0 and max_depth=2). This bagging classifier should **keep all training instances but sample features without replacement (maximum features = 40)**. Use random_state=0 for bagging model as well.

(d) (10 points) Bagging doesn't overfit as the number of models are increased. We should find the optimal number of models in your bagging model in part c. Starting from 1, increase the number of models by 1 until 40. Calculate test accuracy (although you would ideally use validation accuracy calculated with validation set/cross validation, report test accuracy calculated with test test in this assignment) of each bagging model. Plot test accuracy of bagging models as shown in the plot below. Although there could be local ups and downs, general trend will be an increase in test accuracy until it converges.



(e) (4 points) What is the best number of models (in the range of 1 to 40)? (If there are several best numbers with same accuracy, report the highest number). Calculate training and test accuracy of the bagging model for the best number of models.

(f) (6 points) Compare the decision tree classifier in part a and best bagging classifier in part e in terms of accuracy, bias, and variance.

(g) (5 points) Create a bagging classifier with decision tree classifier (with parameters of random_state=0 and max_depth=2). This bagging classifier should **sample training instances with replacement (maximum samples=1.0) and sample features without replacement (maximum features = 40)** Use random_state=0 for bagging model.

(h) (10 points) Calculate test accuracy of bagging model you created in part g as you increase the number of models from 1 to 40 . Plot test accuracy of these bagging models as in part d.

(i) (4 points) What is the best number of models in part h? (If there are several best numbers with same accuracy, report the highest number). Calculate training and test accuracy of the bagging model for the best number of models.

(j) (6 points) Compare the decision tree classifier in part a and best bagging classifier in part i in terms of accuracy, bias, and variance.

(k) (5 points) Compare two bagging models that you found in part e and part i in terms of accuracy, bias, and variance.

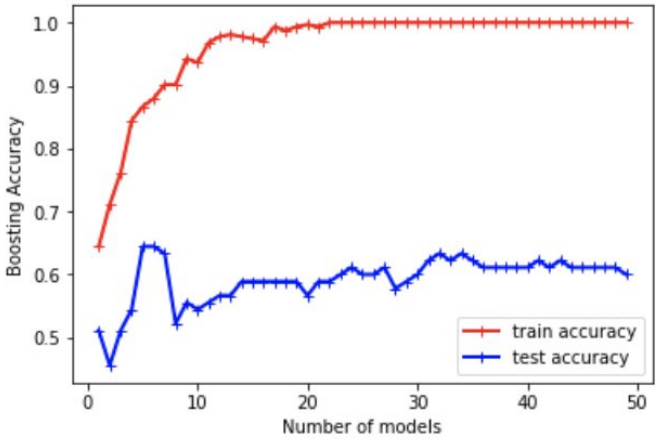
(l) (5 points) How sampling features and sampling instances affected the performance of your model? Why?

BOOSTING (15 points)

Boosting is an ensemble method that can combine several weak learners (very simple models such as Decision Stump) into a strong learner.

(m) (2 points) Create an AdaBoost classifier with decision trees (with parameters of random_state=0 and max_depth=2). Use random state=0 for the boosting model.

(n) (10 points) Unlike Bagging, Boosting can lower both Bias & Variance and it can potentially overfit. We should find the optimal number of models in your boosting model. Starting from 1, increase the number of models by 1 until 50. Calculate train and test accuracy of each boosting model. Plot train and test accuracy of boosting models as shown in the plot below.



(o) (3 points) What is the best number of models? (If there are several best numbers with same accuracy, report the highest number). Report train and test accuracy of the boosting model which has the best number of models.

STACKING (20 points)

(p) (5 points) Split **X_train** and **y_train** into two sets using train_test_split function with parameters of random_state=0, test_size=0.5. Name resulting data sets as **train_set_1, train_set_2** and **train_y_1, train_y_2** respectively.

Create Logistic Regression, and SVC with parameters in the table below. Train individual classifiers with **train_set_1** and **train_y_1**.

SVC	Logistic Regression
‘random_state’ : 0, 'C': 0.1, 'decision_function_shape': 'ovo', 'kernel': 'poly', 'tol': 0.0001, ‘probability’: True	‘random_state’ : 0, 'C': 0.001, 'penalty': 'l2', ‘solver’: ‘saga’

(q) (10 points) Run individual classifiers from part p to make **probabilistic predictions** on the **train_set_2** and then create a **new_train_set_2** with the resulting predictions: each instance is a vector containing the set of probabilistic predictions from all your classifiers for an image, and the target is the image’s class.

For instance, for each image, firstly, you will predict the probability vector of the image with your individual classifiers' predict_proba() function. Then take only the first probability.
Let's say Logistic Regression classifier's probabilistic prediction output is: [0.2, 0.8], SVC classifier's probabilistic prediction output is: [0.3 0.7].
The new representation of the image will be: [0.2, 0.3] and the label will stay as the original label.

Create a RandomForest classifier as a **blender** with parameters of random state=0, max_depth=1, n_estimators=10. Train this model with **new_train_set_2** and **train_y_2**.

(r) (5 points) Use individual classifiers from part p to create a **new_test_set** with the probabilistic predictions of **X_test** as you did in part q **(** Do not fit models into test data, just perform prediction by using the models from part p). Run RandomForest blender classifier (that you trained in part q) on **new_test_set** to get final predictions of your test model. Report accuracy and f1 scores of new_test_set. Did stacking model perform well?

IMPORTANT NOTES

- Prepare and upload one Jupyter notebook file which should be named as <your first name>_<your last name>_ assignment3.ipynb.
- A sample Jupyter notebook file provided to you. Follow the template's structure.
- **Explain your code with comments. If you don't explain your code with reasonable comments, you will get 0 from corresponding part.**

Wrong file name format	-10 points
Not using template	-10 points