MLDL Assignment 2

작성자 2022-24670 송종현

1. **In this problem, you will use the OnlineAd training data set attached in the assignment (OnlineAd X train.csv and OnlineAd Y train.csv). The feature matrix OnlineAd X train.csv contains 1452 observations, where each row represents a different user with 251 features that summarize the user characteristics and previous browsing history. These 251 features have been anonymized and normalized. Hence, each feature's interpretation is hidden for analysis. (i.e., you don't have to worry about the meaning of each feature.) There are two online advertisements, A and B, with which this experiment was performed. In each observation, a user clicks either A or B, or she may not click anything. She cannot click on two ads at the same time. This click response is recorded in OnlineAd Y train.csv, where each row contains information on which ad was clicked (or whether a user clicked nothing). A no click response is recorded as 1 in the first column in OnlineAd Y train.csv, a click on A is 1 in the second column, and likewise for a click on B is 1 in the third column. Note that 0 means non-chosen options by the user. Hence, each row sum of OnlineAd Y train.csv is exactly 1. Each row of OnlineAd Y train.csv is the click response for the corresponding user in the corresponding row of OnlineAd X train.csv**

   (a) [40 pts] Using OnlineAd X train.csv and OnlineAd Y train.csv, train multiple models that you learned in class. You may try a model which is modified from the models we covered in the first half of the course. You are allowed to use the existing packages, but make sure to clearly explain what models are used. Report the training results. What metric did you use? How do different models perform on the training data?

   Given the multi-class classification problem, modification 'Y_train.csv' into the pandas dataframe whose label is '1' for No click, '2' for 'Click on A', and '3' for 'Click on B' was carried out.

```
#import CSV file
X_train = pd.read_csv('OnlineAd_X_train.csv', header=None)
Y_train = pd.read_csv('OnlineAd_Y_train.csv',names= ['class','class 2','class 3'])
Y_train.loc[Y_train['class 2'] == 1, ['class']] = 2
Y_train.loc[Y_train['class 3'] == 1, ['class']] = 3
Y_train.drop(columns= ['class 2','class 3'])
Y_train= Y_train['class']
Y_train=pd.DataFrame(Y_train)
print(Y_train)
```

To figure out the characteristic of a given dataset, principal component analysis (PCA) was implemented. However, the given dataset is inappropriate for PCA as a few variances were explained by PCA. (Details will be covered later on)

All the classification methods which are learned from the class- Logistic Regression, K-Nearest Neighbor, Discriminant Analysis, Decision Tree, Bagging, Boosting, Random Forests, and Support Vector Machine- were carried out. To figure out the training results, cross-validation using the package RepeatedStratifiedKFold() was performed. (its default fold is 5 and repeats are 10. I followed the default settings.) the metric was accuracy from cross-validation.

(1) Logistic Regression

```
# Most of code blocks below were referenced from sckitlearn manual.
logreg=LogisticRegression(max_iter=1000)
Y_train=np.ravel(Y_train)
cv=RepeatedStratifiedKFold()
scores= cross_val_score(logreg,X_train, Y_train, cv=cv)
est = smd.MNLogit(Y_train.ravel(), X_train).fit()
print(np.mean(scores))
print(np.std(scores))
print(scores)
```

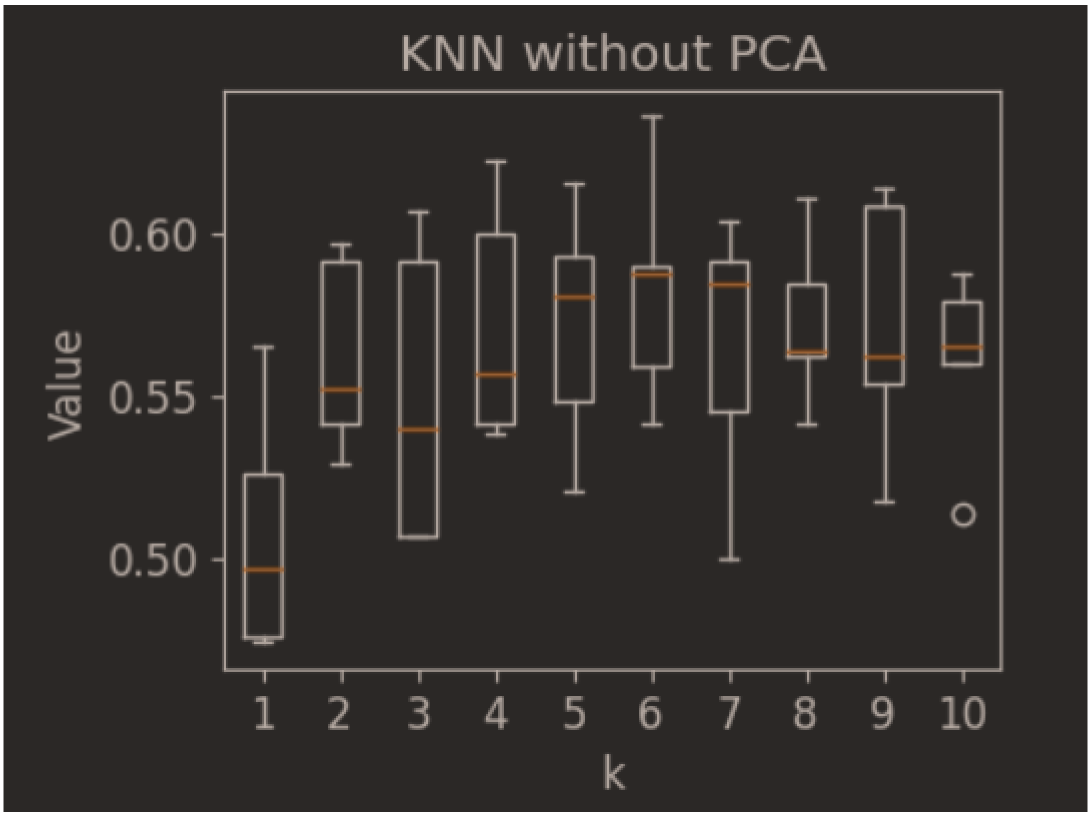
The code above showed that the mean CV score was 0.55177(±0.0231), which is not that different from flipping coins. The reason for low accuracy is given dataset is highly non-linear, which can be checked from principal component analysis which will be covered soon.

(2) K-nearest Neighbor

```
knn_results=[[None for x in range (2)]for y in range (10)]
for i in range(1,11):
    clf = KNeighborsClassifier(n_neighbors= i )
    scores= cross_val_score(clf,X_train, Y_train, scoring='accuracy', cv=5)
    knn_results[i-1][0]=scores
plt.style.use('default')
plt.rcParams['figure.figsize'] = (4, 3)
plt.rcParams['font.size'] = 12
fig, ax = plt.subplots()
ax.boxplot([knn_results[0][0],knn_results[1][0],knn_results[2][0],knn_results[3]
[0],knn_results[4][0],knn_results[5][0],knn_results[6][0],knn_results[7][0],knn_re
sults[8][0],knn_results[9][0]])
ax.set_xlabel('k')
ax.set_title('KNN without PCA')
ax.set_ylabel('Value')
plt.show()
fig, ax = plt.subplots()
ax.boxplot([knn_results[0][1],knn_results[1][1],knn_results[2][1],knn_results[3]
[1],knn_results[4][1],knn_results[5][1],knn_results[6][1],knn_results[7][1],knn_re
sults[8][1],knn_results[9][1]])
ax.set_xlabel('k')
ax.set_title('KNN with PCA')
ax.set_ylabel('Value')
plt.show()
```

(NOTE: cross-validation whose fold is 5 instead of RepeatedStratifiedFold() package was used due to a computing problem)

To choose which hyperparameter k is the best, cross-validation was performed for each k from 1 to 10. The result, which is represented as a box-plot, is as below



Like linear regression, there was no big difference with flipping coins.

(3) LDA & QDA

As there was an error message that variables are collinear, PCA whose number of principal is three was used for predictors.

```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis

results=[[None for x in range (2)]for y in range (2)]
lda = LinearDiscriminantAnalysis()
qda = QuadraticDiscriminantAnalysis(store_covariance=True)
cv=RepeatedStratifiedKFold()
scores_lda= cross_val_score(lda,principaldf, Y_train,scoring='accuracy', cv=cv)
scores_qda=cross_val_score(qda,principaldf, Y_train,scoring='accuracy', cv=cv)
print(np.mean(scores_lda))
print(np.mean(scores_qda))
print(np.std(scores_lda))
print(np.std(scores_qda))

print(scores_lda)
print(scores_qda)
```

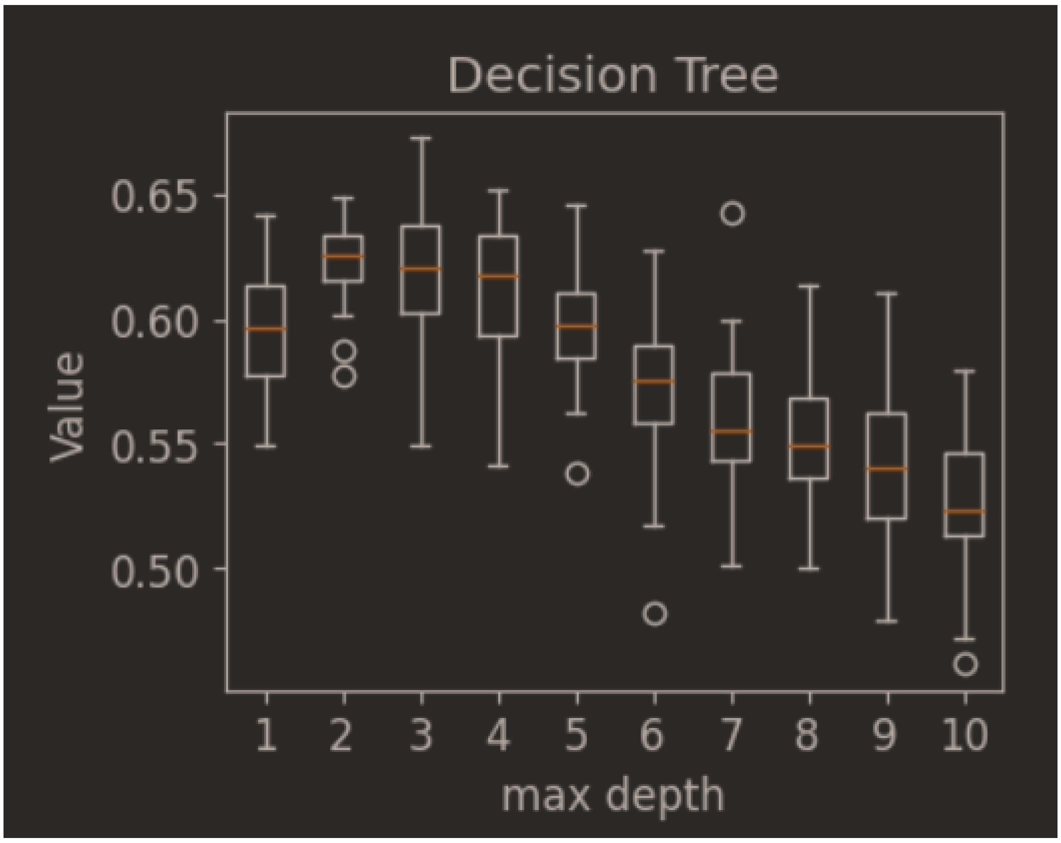
The performance of LDA and QDA was 0.647(±0.0167) and 0.641(±0.0218) for each.

As the variance of each predictor must be different, It is hard to understand the accuracy of LDA is higher than that of QDA. however, as overall accuracy are too low (<70%), a comparison between them could be meaningless. Details on variable reduction techniques such as PCA will be covered later on.

(4) Decision Trees

To determine which decision tree is the best, cross-validation on different max depths was carried out. The code and result are as below.

```
from sklearn import tree
df_results=[None for y in range (10)]
for i in range(1,11):
    clf = tree.DecisionTreeClassifier(criterion='entropy', max_depth=i)
    cv = RepeatedStratifiedKFold()
    scores= cross_val_score(clf,X_train, Y_train, scoring='accuracy', cv=cv)
    df_results[i-1]=scores
plt.style.use('default')
plt.rcParams['figure.figsize'] = (4, 3)
plt.rcParams['font.size'] = 12
fig, ax = plt.subplots()
ax.boxplot([df_results[0],df_results[1],df_results[2],df_results[3],df_results[4],
df_results[5],df_results[6],df_results[7],df_results[8],df_results[9]])
ax.set_xlabel('max depth')
ax.set_title('Decision Tree')
ax.set_ylabel('Value')
plt.show()
```



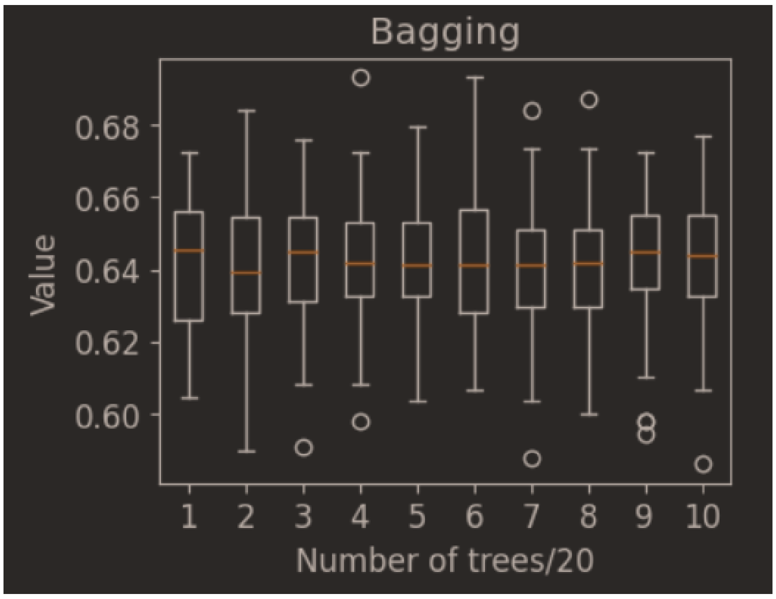
the maximum average CV accuracy was <65%. It means that additional techniques such as bagging, boosting, or random forests are required.

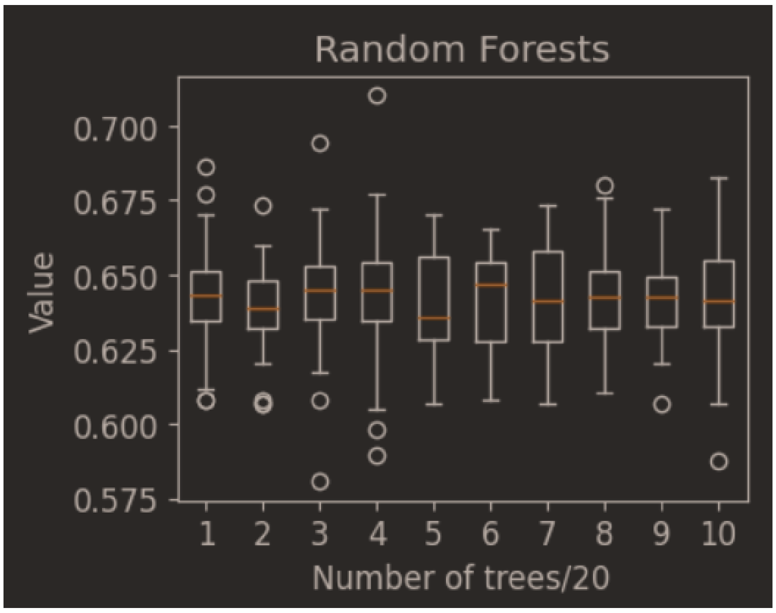
(5) Bagging, Boosting, Random Forests

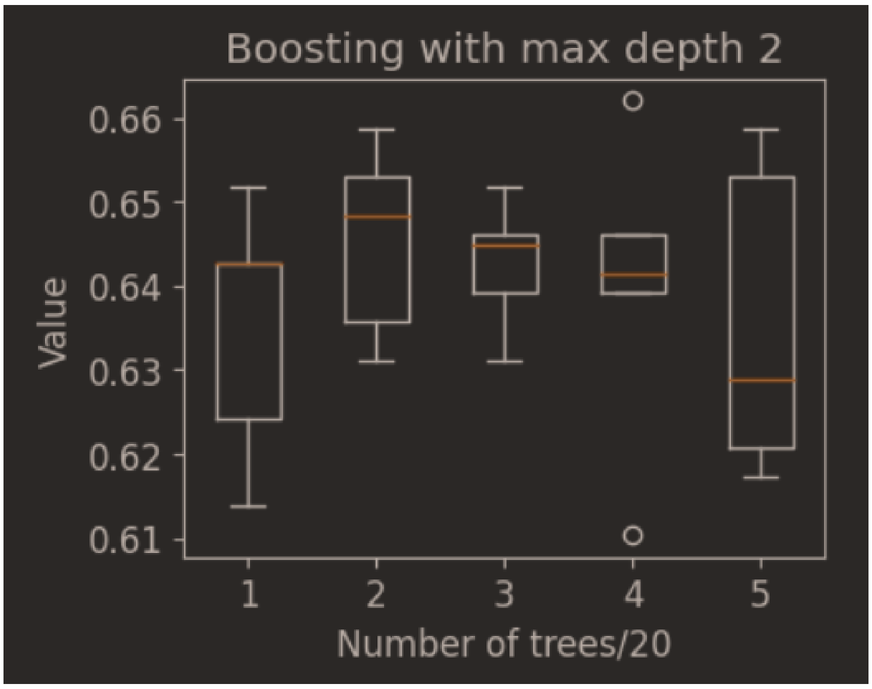
```
#Bagging
from sklearn.ensemble import BaggingClassifier
df_results=[None for y in range (10)]
for i in range(1,11):
    model = BaggingClassifier(n_estimators=20*i)
    cv = RepeatedStratifiedKFold()
    scores= cross_val_score(clf,X_train, Y_train, scoring='accuracy', cv=cv)
    df_results[i-1]=scores
plt.style.use('default')
plt.rcParams['figure.figsize'] = (4, 3)
plt.rcParams['font.size'] = 12
fig, ax = plt.subplots()
ax.boxplot([df_results[0],df_results[1],df_results[2],df_results[3],df_results[4],
df_results[5],df_results[6],df_results[7],df_results[8],df_results[9]])
ax.set_xlabel('Number of trees/20')
ax.set_title('Bagging')
ax.set_ylabel('Value')
plt.show()

#Random Forests
from sklearn.ensemble import RandomForestClassifier
```

```
df_results=[None for y in range (10)]
for i in range(1,11):
    model = RandomForestClassifier(n_estimators=20*i, max_features='sqrt')
    cv = RepeatedStratifiedKFold()
    scores= cross_val_score(clf,X_train, Y_train, scoring='accuracy', cv=cv)
    df_results[i-1]=scores
plt.style.use('default')
plt.rcParams['figure.figsize'] = (4, 3)
plt.rcParams['font.size'] = 12
fig, ax = plt.subplots()
ax.boxplot([df_results[0],df_results[1],df_results[2],df_results[3],df_results[4],
df_results[5],df_results[6],df_results[7],df_results[8],df_results[9]])
ax.set_xlabel('Number of trees/20')
ax.set_title('Random Forests')
ax.set_ylabel('Value')
plt.show()

#Boosting
from sklearn.ensemble import GradientBoostingClassifier
Y_train=np.ravel(Y_train)
df_results=[None for y in range (10)]
for i in range(1,11):
    model = GradientBoostingClassifier(n_estimators=20*i, max_depth=2,criterion='s
quared_error')
    cv = RepeatedStratifiedKFold()
    scores= cross_val_score(model,X_train, Y_train, scoring='accuracy', cv=cv)
    df_results[i-1]=scores
plt.style.use('default')
plt.rcParams['figure.figsize'] = (4, 3)
plt.rcParams['font.size'] = 12
fig, ax = plt.subplots()
ax.boxplot([df_results[0],df_results[1],df_results[2],df_results[3],df_results[4],
df_results[5],df_results[6],df_results[7],df_results[8],df_results[9]])
ax.set_xlabel('Number of trees/20')
ax.set_title('Boosting with max depth 2')
ax.set_ylabel('Value')
plt.show()
```

Box plots for each method by number of trees are as below.


Bagging
Value
0.68
0.66
0.64
0.62
0.60
1
2
3
4
5
6
7
8
9
10
Number of trees/20


Random Forests
Value
0.700
0.675
0.650
0.625
0.600
0.575
1
2
3
4
5
6
7
8
9
10
Number of trees/20

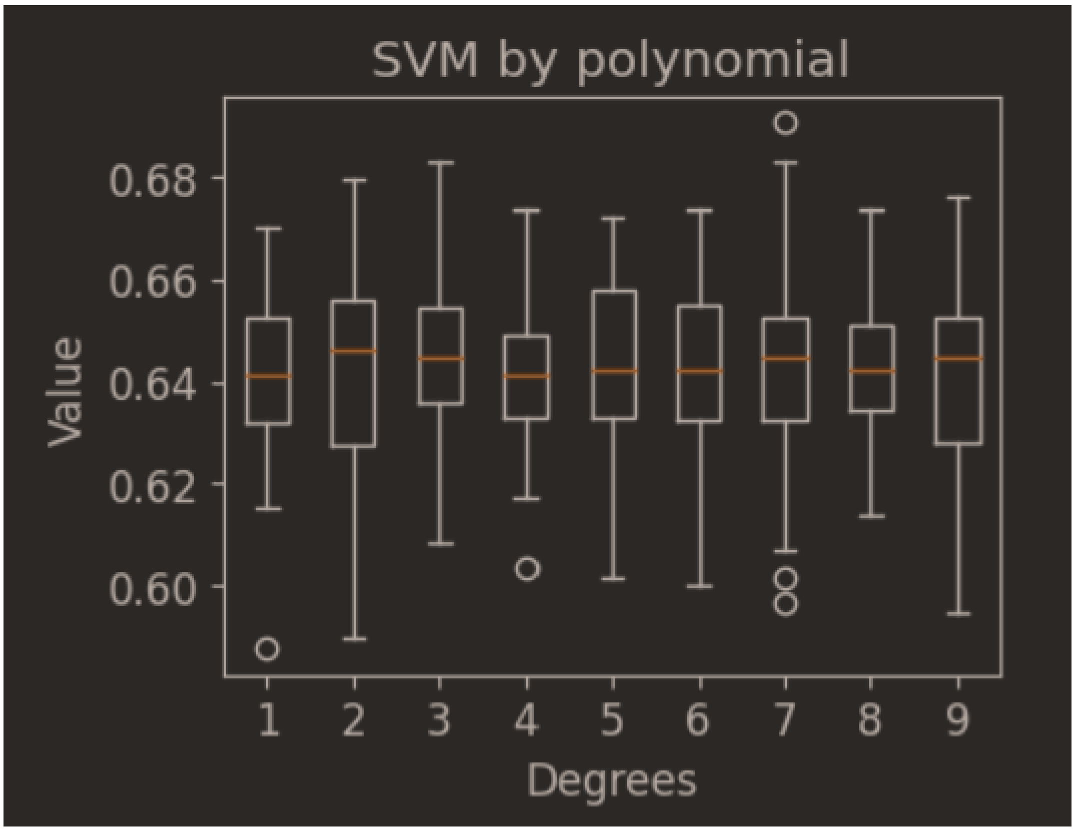


(6) Support Vector Machine

support vector machines using a variety of kernels - polynomial and radial- were used and verified by cross-validations.

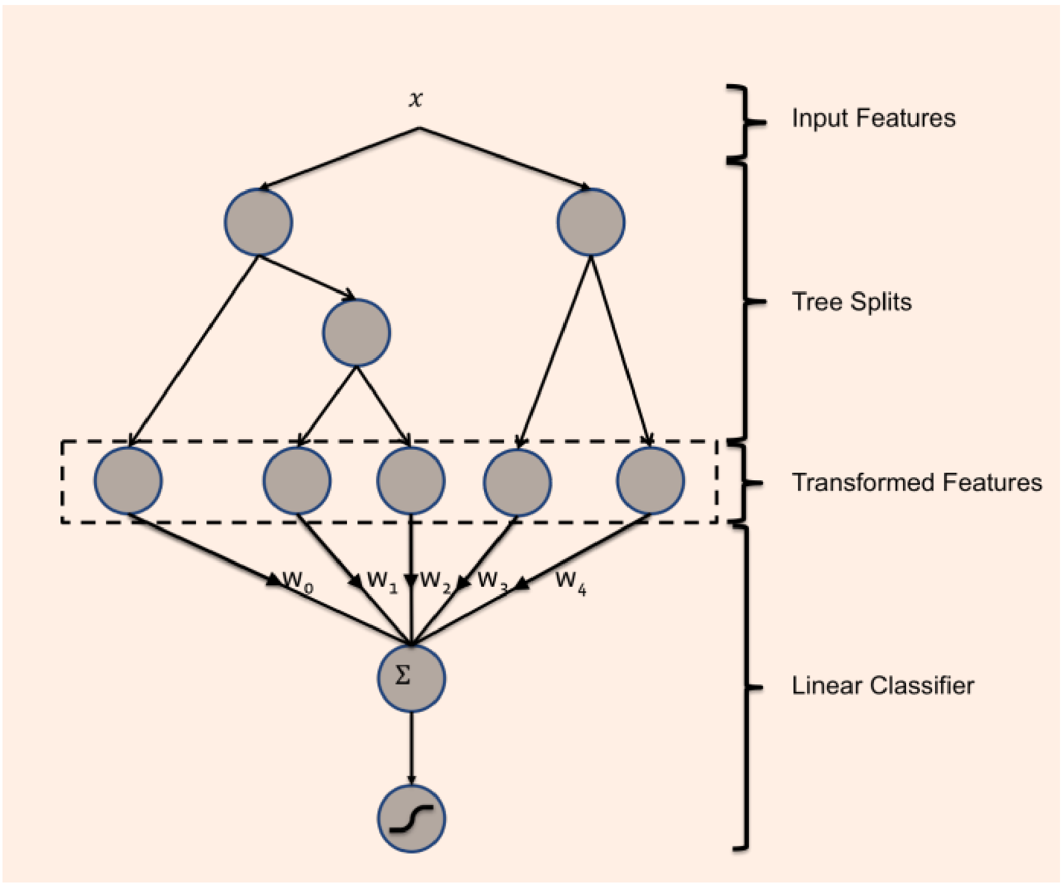
```
from sklearn import svm

clf = svm.SVC(kernel='rbf')
clf.fit(X_train, Y_train)
cv = RepeatedStratifiedKFold()
scores= cross_val_score(clf,X_train, Y_train, scoring='accuracy', cv=cv)
print("rbf")
print(np.mean(scores))
print(np.std(scores))

df_results=[None for y in range (9)]
for i in range(1,11):
    clf = svm.SVC(degree=i)
    cv = RepeatedStratifiedKFold()
    scores= cross_val_score(clf,X_train, Y_train, scoring='accuracy', cv=cv)
    df_results[i-1]=scores
plt.style.use('default')
plt.rcParams['figure.figsize'] = (4, 3)
plt.rcParams['font.size'] = 12
fig, ax = plt.subplots()
ax.boxplot([df_results[0],df_results[1],df_results[2],df_results[3],df_results[4],
df_results[5],df_results[6],df_results[7],df_results[8]])
ax.set_xlabel('Degrees')
ax.set_title('SVM by polynomial')
ax.set_ylabel('Value')
plt.show()
```

The Cross-validation accuracy of the radial kernel is 0.643(±0.0201) and the polynomials is as below:



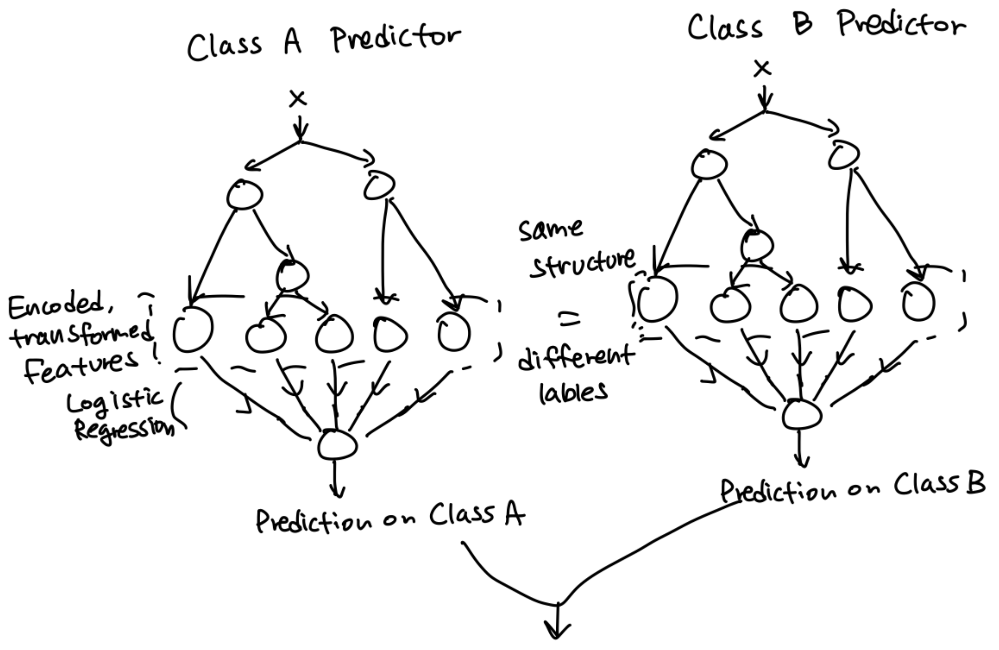
(7) A model which combines decision trees with logistic regression

Sadly, models presented in (1)~(6) showed terrible prediction accuracy. For this challenging task, the model which has been introduced in the paper "Practical lessons from predicting clicks on ads at facebook." (He, Xinran, et al. , 2014) was implemented to solve the problem. This model showed 74~78% prediction accuracy, which is a drastic improvement considering that there were no models whose prediction accuracy was higher than 65%.



To briefly explain, non-linear input features are simplified into categorical features by a gradient boosting machine. Then transformed features, which are 1-of-K coded, are treated as an input feature to the logistic regression model. By doing so, we can get the advantage of learning linear models by non-linear input features.

As the given model is a binomial model, training data for class A and class B was made use of independently. If both models predict 'Yes' for the same predictor, then the class with higher probability was chosen as multiple choices are not allowed.



Four Cases

Class A	Class B	
No	No	First column=1
Yes	No	Second column=1
No	Yes	Third column=1
Yes	Yes	If probability for class A is higher → Second column=1 Otherwise, third column=1

```
# Changing given dataset into binary classification
# partially referred to https://github.com/asirzhao/Machine-learning-demo/blob/mas
ter/algorithm/gbtWithLogisticRegression/gradient_logistic.py
Y_train_tmp=pd.read_csv('OnlineAd_Y_train.csv',names= ['No click','class A','class
B'])
Y_train_A=Y_train_tmp['class A']
Y_train_B=Y_train_tmp['class B']
M_train_A=pd.concat([X_train,Y_train_A], axis=1)
M_train_B=pd.concat([X_train,Y_train_B], axis=1)

#model A training with 5-fold cross-validation
from sklearn.ensemble import GradientBoostingClassifier
import sklearn.preprocessing as pp

M_train_A_randomized=M_train_A.sample(frac=1)
for count in range(5):
    train = M_train_A_randomized.iloc[290*count:290*(count+1),:]
    test= pd.concat([M_train_A_randomized.iloc[:290*count,:],M_train_A_randomized.
iloc[290*(count+1):,:]])
    model = GradientBoostingClassifier(max_depth=2, criterion='squared_error')
    X_train_set=train.iloc[:,0:251]
    Y_train_set=train["class A"]
    X_test_set=test.iloc[:,0:251]
    Y_test_set=test["class A"]
    model.fit(X_train_set,Y_train_set)
    leaves=pd.DataFrame(model.apply(X_train_set)[:,:,0])
    enc=pp.OneHotEncoder().fit(leaves)
    x_enc=enc.transform(leaves)
    logreg=LogisticRegression(max_iter=1000).fit(x_enc, Y_train_set)
    leaves2=pd.DataFrame(model.apply(X_test_set)[:,:,0])
    X_set2_enc=enc.transform(leaves2)
    print(logreg.score(X_set2_enc, Y_test_set))

#model B training with 5-fold cross-validation
M_train_B_randomized=M_train_B.sample(frac=1)
for count in range(5):
    train = M_train_B_randomized.iloc[290*count:290*(count+1),:]
    test= pd.concat([M_train_B_randomized.iloc[:290*count,:],M_train_B_randomized.
iloc[290*(count+1):,:]])
    model = GradientBoostingClassifier(max_depth=2, criterion='squared_error')
    X_train_set=train.iloc[:,0:251]
    Y_train_set=train["class B"]
    X_test_set=test.iloc[:,0:251]
    Y_test_set=test["class B"]
    model.fit(X_train_set,Y_train_set)
    leaves=pd.DataFrame(model.apply(X_train_set)[:,:,0])
    enc=pp.OneHotEncoder().fit(leaves)
    x_enc=enc.transform(leaves)
    logreg=LogisticRegression(max_iter=1000).fit(x_enc, Y_train_set)
    leaves2=pd.DataFrame(model.apply(X_test_set)[:,:,0])
    X_set2_enc=enc.transform(leaves2)
    print(logreg.score(X_set2_enc, Y_test_set))
```

The prediction accuracy of the former is 0.767(±0.0144) and the latter is 0.781(±0.0144). Quite High compared to previous results.

(b) [20 pts] Do you think dimension reduction on features (or feature selection) is needed here? If so, provide analysis on which features may be important. If not, please justify your answer.

A Couple of methods can be considered.

The first one is neglecting the variable with the lowest variance (Choosing the predictors with higher variance) Code below is for finding 50 predictors with the highest or lowest variances.

```
# Finding Variance

df_var= pd.DataFrame(X_train.var())
```

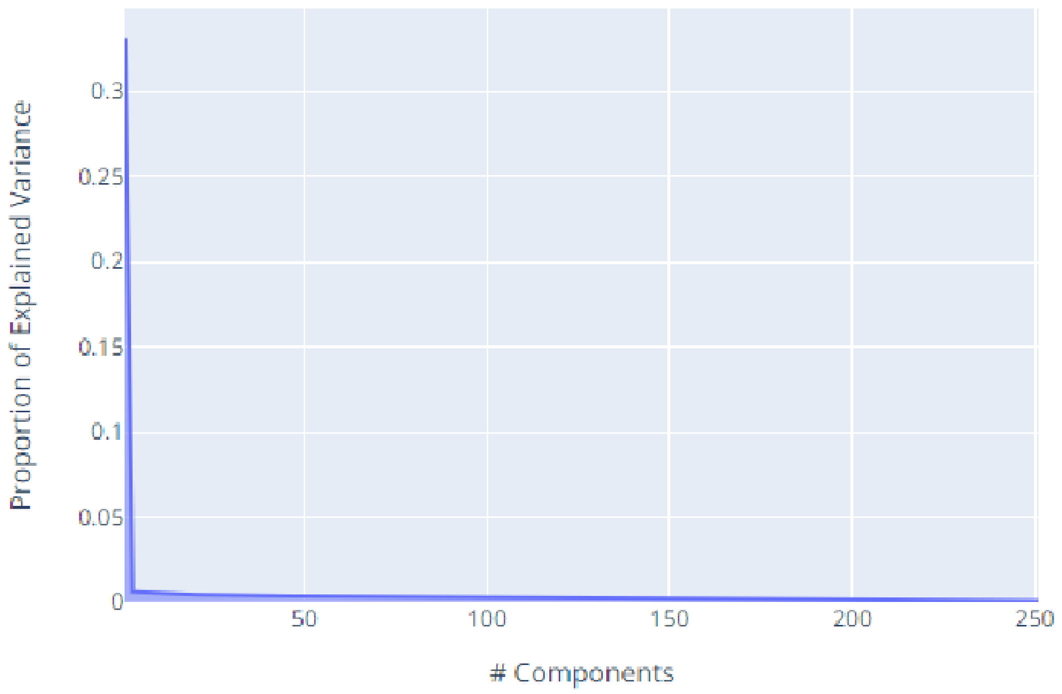
```
print(df_var[0].nsmallest(n=50))
print(df_var[0].nlargest(n=50))
```

However, the lowest predictor had a variance of 0.303, which is no big difference from 0.399, the lowest variance amongst the 50 highest variances. In conclusion, the variances of predictors show very dense population and there is no reason to ignore the variables with lower variance.

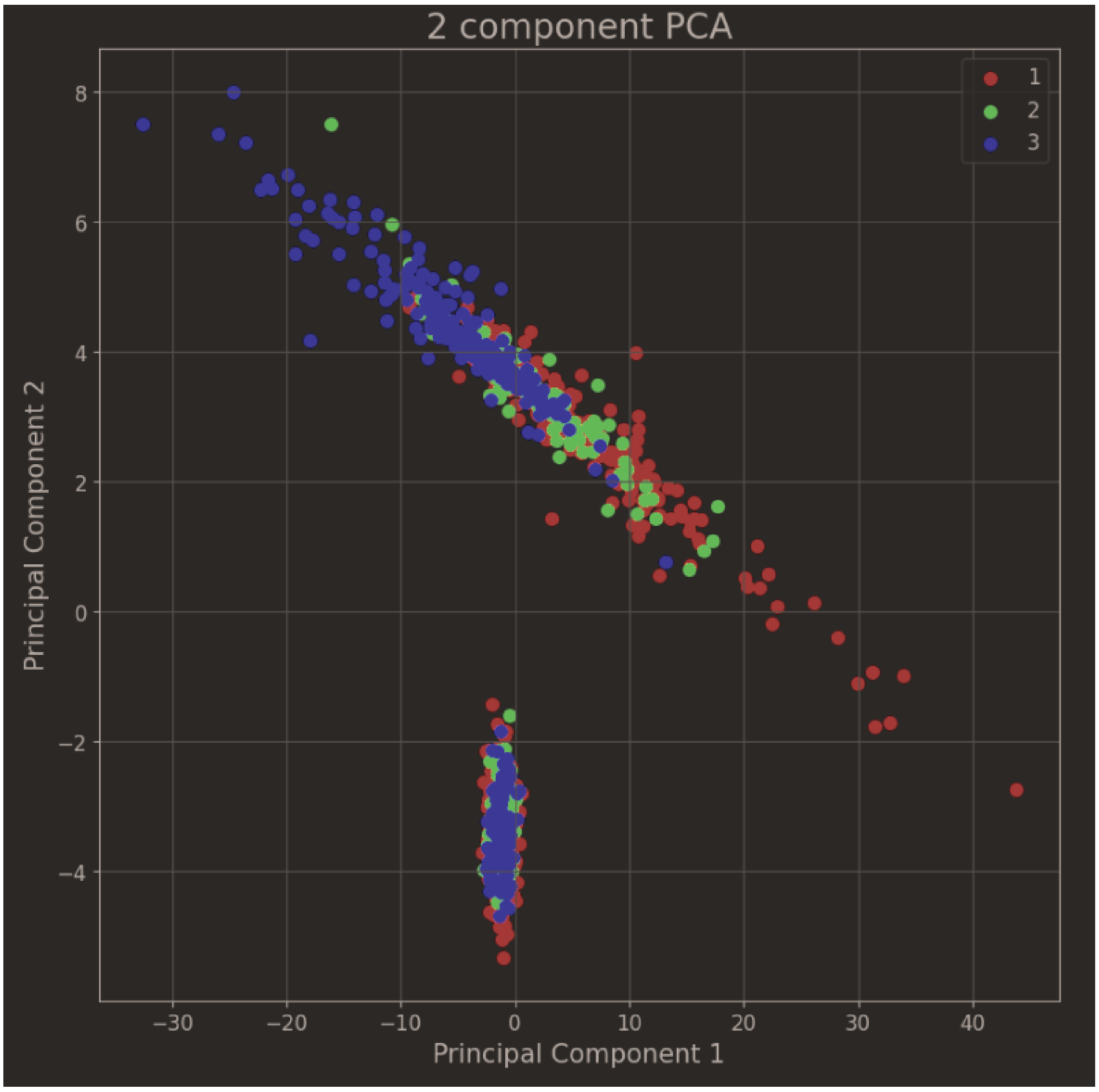
The second one is principal component analysis, which is the most famous methodology for dimension reduction. PCA can be carried through as code below.

```
# Plotting Scree plot.
# Source: https://m.blog.naver.com/tjdrud1323/221720259834
pca = PCA()
pca.fit(X_train)
exp_var =pca.explained_variance_ratio_
px.area(
    x=range(1, exp_var_cumul.shape[0] + 1),
    y=exp_var,
    labels={"x": "# Components", "y": "Proportion of Explained Variance"}
)
```

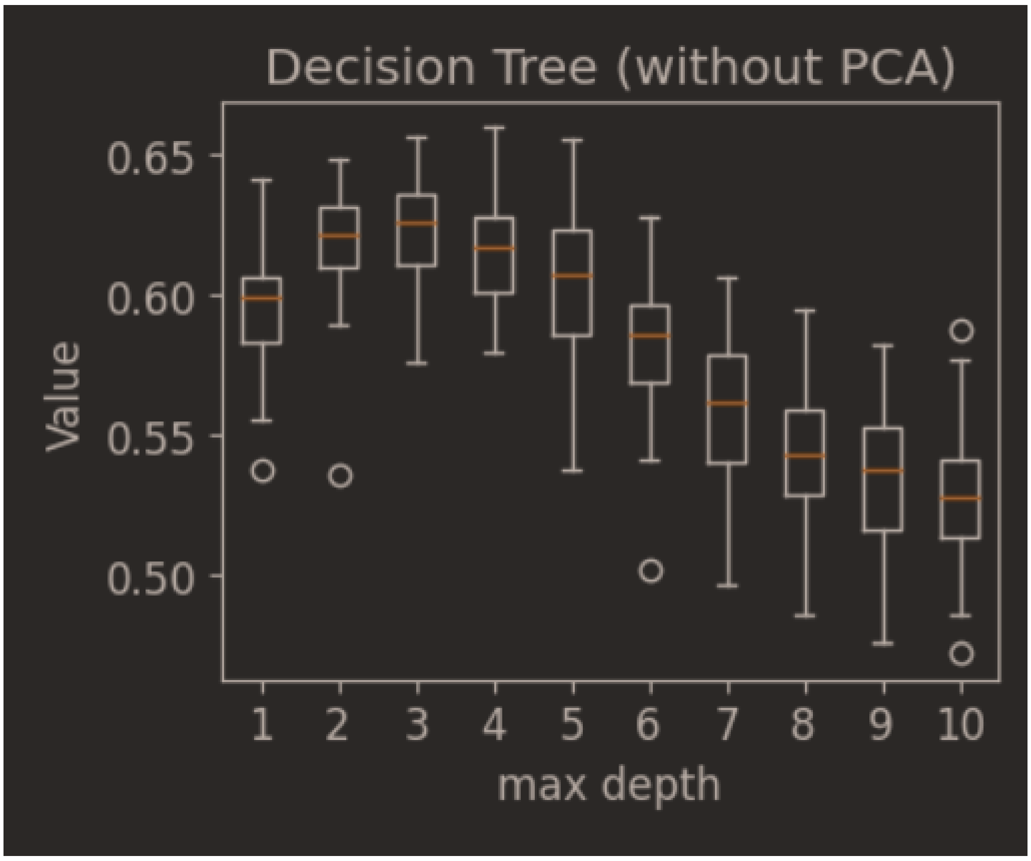
The Scree plot is as below. principal axes explain 0.33, 0.14, 0.006... of the total variance. Even though the elbow of the scree plot is at the third principal axis, those have limitations that they are even not able to explain half of the total variance. (cumulative variance is lower than 0.5)

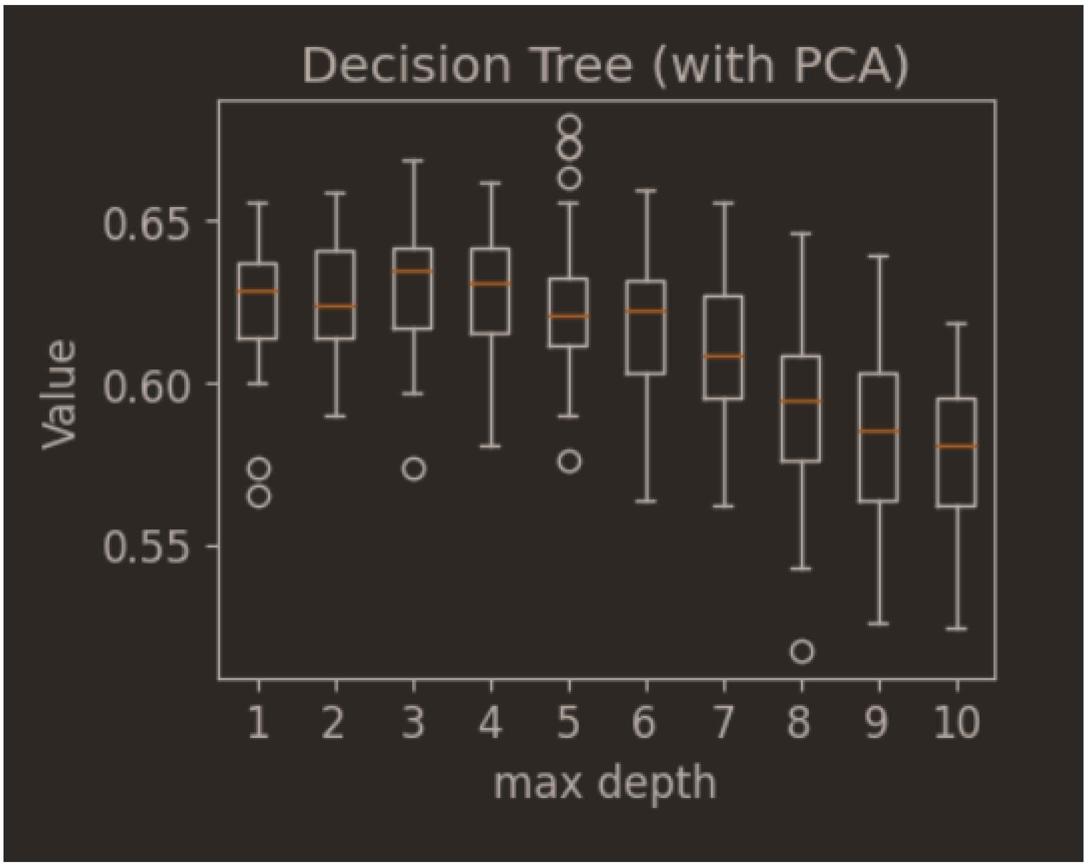


Additional visualization is as below. It is figured out that PCA has limitations in separating given datasets. (especially in the island-like-area at the bottom side)



Does PCA have an effect on improving prediction accuracy?


Decision Tree (without PCA)
Value
0.65
0.60
0.55
0.50
1
2
3
4
5
6
7
8
9
10
max depth


Decision Tree (with PCA)
Value
0.65
0.60
0.55
1
2
3
4
5
6
7
8
9
10
max depth

Above are box plots for decision tree with or without PCA. there's no big difference on prediction accuracy by dimension reduction.

Because of reasons above, dimension reduction is not prerequisite for solving the problem.

2. **Based on your training results in Problem 1, you now pick your best model that would generalize well to unseen data. Using your best model, we are going to predict on the provided test dataset, OnlineAd X test.csv which contains 300 observations with 251 features. Here we do not have the corresponding responses for these test observations.**

(a) [20 pts] Report the estimated test performance for your best model. Provide a reason for your choice of a model among the models you considered.

The prediction accuracy of the best model was around 74~79% from 5-fold cross-validation.

There are some reasons to choose Trees-Logistic Regression combined model.

- *Only using one model showed obvious drawbacks. Even some models had no difference with flipping coins! This might be because given datasets do not satisfy the assumptions required to use specific models. It can be overcome by combining models.*
- *The model was initially proposed from the paper about the clicking ad prediction problem on Facebook, which means predictors and labels share similar characteristics.*
- *Given datasets are highly complicated. Transforming input features into encoded vectors by trees decreases the difficulty of predicting from non-linear datasets.*

(b) [20 pts] Predict on the provided test dataset, OnlineAd X test.csv, and save those predictions as a CSV file named [your-student-ID] pred.csv. The CSV file should only contain the array of dimension [300, 3] in the same format as the OnlineAd Y train.csv except the number of rows, since there are only 300 observations in the test data, i.e., the first column corresponds to no click, the second column corresponds to the ad A, and the third column corresponds to the ad B. A violation of this format guideline will result in 10 point penalty.

F*iles are attached as instructions.*

(c) [Extra 10 pts] For those whose submissions achieve the top 5 test performances out of all the submissions, an extra credit will be given.

I hope I could win.