In [59]:
```
import pandas as pd
from matplotlib import pyplot as plt
%matplotlib inline

import numpy as np
np.random.seed(13)

import tensorflow as tf
import pandas as pd
from tensorflow.keras import backend as K
from tensorflow.keras.models import Sequential
from tensorflow.keras.datasets import mnist
from tensorflow.keras.layers import Dense, Activation, Dropout
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.wrappers.scikit_learn import KerasClassifier

import matplotlib.pyplot as plt
import scikitplot as skplt
from sklearn import neighbors, datasets, tree, linear_model, metrics,svm
from sklearn.model_selection import cross_val_score, train_test_split,KFold
import itertools
from itertools import permutations
from sklearn.metrics import recall_score, confusion_matrix, mean_squared_error
from sklearn.model_selection import GridSearchCV
```

1. Import data

In [60]:
```
df=pd.read_excel('HW3.xlsx')
df=df.drop(columns=['sequence_number','Purchase'])
```

2. Formating the data

In [61]:
```
c=['US', 'source_a', 'source_c', 'source_b', 'source_d', 'source_e','source_m', 'source_o', 'source_h',
 'source_r',
   'source_s', 'source_t','source_u', 'source_p', 'source_x', 'source_w','Web order','Gender=male', 'Ad
dress_is_res']
df[c] = df[c].astype(str)
```

In [62]:
```
df.dtypes
```

Out[62]:
```
US                     object
source_a               object
source_c               object
source_b               object
source_d               object
source_e               object
source_m               object
source_o               object
source_h               object
source_r               object
source_s               object
source_t               object
source_u               object
source_p               object
source_x               object
source_w               object
Freq                    int64
last_update_days_ago    int64
1st_update_days_ago     int64
Web order              object
Gender=male            object
Address_is_res         object
Spending              float64
dtype: object
```

(a)

In [63]:
```
#Split Train and Test Data
x_train,x_test,y_train,y_test = train_test_split(df.drop(columns=['Spending']),df['Spending'],test_size
=0.2,random_state=9)

#Standardization
from sklearn.preprocessing import MinMaxScaler
mms = MinMaxScaler()
mms.fit(x_train[['Freq','last_update_days_ago','1st_update_days_ago']])
x_train[['Freq','last_update_days_ago','1st_update_days_ago']]=mms.transform(x_train[['Freq','last_upda
te_days_ago','1st_update_days_ago']])
x_test[['Freq','last_update_days_ago','1st_update_days_ago']]=mms.transform(x_test[['Freq','last_update
_days_ago','1st_update_days_ago']])
```

In [65]:
```
#Elastic Net Regression
from sklearn.linear_model import ElasticNet
grid={'alpha':[1,2,3,4,5,6,7,8,9,10],
      'l1_ratio':[0.3,0.5,0.7]}
EN_linear=ElasticNet(random_state=99)
model = GridSearchCV(estimator=EN_linear, param_grid=grid, cv=5,scoring='neg_mean_squared_error')
model.fit(x_train,y_train)
print("With CV grid search, I found the best hyperparameter is alpha={} and L1 ratio={}.".format(model.
best_params_['alpha'],model.best_params_['l1_ratio']))
print("MSE on Test Data: {}".format(round((metrics.mean_squared_error(y_test, model.predict(x_test))),2
)))
```

```
With CV grid search, I found the best hyperparameter is alpha=1 and L1 ratio=0.7.
MSE on Test Data: 31795.78
```

In [66]:
```
#KNN regression
from sklearn.neighbors import KNeighborsRegressor
grid={'n_neighbors':[5,10,15,20,50,100,500],
      'weights':['uniform','distance']}
KNN=KNeighborsRegressor()
model = GridSearchCV(estimator=KNN, param_grid=grid, cv=5,scoring='neg_mean_squared_error')
model.fit(x_train,y_train)
print("With CV grid search, I found the best hyperparameter is n_neighbors={} and weights={}.".format(m
odel.best_params_['n_neighbors'],model.best_params_['weights']))
print("MSE on Test Data: {}".format(round((metrics.mean_squared_error(y_test, model.predict(x_test))),2
)))
```

```
With CV grid search, I found the best hyperparameter is n_neighbors=5 and weights=distance.
MSE on Test Data: 33890.18
```

In [67]:
```
#Regression Tree
from sklearn.tree import DecisionTreeRegressor
grid={'max_depth':[2,3,4,5,6,7,8,9,10,20],
      'min_samples_split':[2,5,10,15,20]}
Tree=DecisionTreeRegressor(random_state=99)
model = GridSearchCV(estimator=Tree, param_grid=grid, cv=5,scoring='neg_mean_squared_error')
model.fit(x_train,y_train)
print("With CV grid search, I found the best hyperparameter is max_depth={} and min_sample_split={}.".f
ormat(model.best_params_['max_depth'],model.best_params_['min_samples_split']))
print("MSE on Test Data: {}".format(round((metrics.mean_squared_error(y_test, model.predict(x_test))),2
)))
```

```
With CV grid search, I found the best hyperparameter is max_depth=7 and min_sample_split=15.
MSE on Test Data: 23458.39
```

In [68]:
```
#SVM Regression
from sklearn.svm import SVR
grid={'kernel':['linear','rbf'],
      'C':[1,2,3,4,5,6,7,8,9,10]}
s=SVR()
model = GridSearchCV(estimator=s, param_grid=grid, cv=5,scoring='neg_mean_squared_error')
model.fit(x_train,y_train)
print("With CV grid search, I found the best hyperparameter is kernel={} and C={}.".format(model.best_p
arams_['kernel'],model.best_params_['C']))
print("MSE on Test Data: {}".format(round((metrics.mean_squared_error(y_test, model.predict(x_test))),2
)))
```

```
With CV grid search, I found the best hyperparameter is kernel=linear and C=10.
MSE on Test Data: 27265.73
```

In [69]:
```
# Random Search With Neural Network
from tensorflow import keras
from tensorflow.keras import layers
from kerastuner.tuners import RandomSearch
```

In [70]:
```
def build_model(hp):
    model = keras.Sequential()
    model.add(layers.Dense(units=hp.Int('units',
                                        min_value=10,
                                        max_value=100,
                                        step=10),
                           activation='relu',input_dim=22))
    model.add(layers.Dense(units=hp.Int('units',
                                        min_value=10,
                                        max_value=100,
                                        step=10),
                           activation='relu'))
    model.add(layers.Dense(1,activation='linear'))
    model.compile(optimizer=keras.optimizers.Adam(hp.Choice('learning_rate',
                                                            values=[0.01,0.001,0.0001])),
                 loss='mse',
                 metrics=['mse'])
    return model

tuner=RandomSearch(build_model,
                   objective='mse',
                   max_trials=3,
                   overwrite=True,
                   executions_per_trial=3)

#Keras cannot input object data type, so no matter the column is boolean or numeric we need to transfor
m them to float32
x_train_float = np.asarray(x_train).astype(np.float32)
y_train_float = np.asarray(y_train).astype(np.float32)
x_test_float = np.asarray(x_test).astype(np.float32)
y_test_float = np.asarray(y_test).astype(np.float32)

tuner.search(x=x_train_float,y=y_train_float,epochs=200,batch_size=32,validation_data=(x_test_float,y_t
est_float))
```

```
Trial 3 Complete [00h 00m 40s]
mse: 18517.412109375

Best mse So Far: 5621.048177083333
Total elapsed time: 00h 01m 58s
INFO:tensorflow:Oracle triggered exit
```

In [72]:
```
result=tuner.get_best_hyperparameters()[0].values
print('The best 3 layers NN parameters would be {} neurons and {} learning rate.'.format(result['units'
],result['learning_rate']))
print('-----------------------------------------')
print("The best model's mse on test data = 5621")
print('-----------------------------------------')
print(tuner.results_summary())
```

```
The best 3 layers NN parameters would be 80 neurons and 0.01 learning rate.
-----------------------------------------
The best model's mse on test data = 5621
-----------------------------------------
Results summary
Results in ./untitled_project
Showing 10 best trials
Objective(name='mse', direction='min')
Trial summary
Hyperparameters:
units: 80
learning_rate: 0.01
Score: 5621.048177083333
Trial summary
Hyperparameters:
units: 70
learning_rate: 0.0001
Score: 18517.412109375
Trial summary
Hyperparameters:
units: 60
learning_rate: 0.0001
Score: 20536.832682291668
None
```

In [73]:
```
#Random Forest Regression
from sklearn.ensemble import RandomForestRegressor
grid={'n_estimators':[100,200,300,400,500],
      'max_depth':[1,2,3,4,5,6,7,8,9,10]}
rf=RandomForestRegressor()
model = GridSearchCV(estimator=rf, param_grid=grid, cv=5,scoring='neg_mean_squared_error')
model.fit(x_train,y_train)
print("With CV grid search, I found the best hyperparameter is n_estimators={} and max_depth={}.".forma
t(model.best_params_['n_estimators'],model.best_params_['max_depth']))
print("MSE on Test Data: {}".format(round((metrics.mean_squared_error(y_test, model.predict(x_test))),2
)))
```

```
With CV grid search, I found the best hyperparameter is n_estimators=100 and max_depth=8.
MSE on Test Data: 23006.77
```

Conclusion:

The **3 layers Neural Network with 80 Neurons** in each layer and **learning rate = 0.01** has the best predictive ability. **The MSE on test data is around 5484**.

(b)

In [76]:
```
df=pd.read_excel('HW3.xlsx')

#Include only the purchase data
df=df[df['Purchase']==1]
df=df.drop(columns=['sequence_number','Purchase'])

#Formating
c=['US', 'source_a', 'source_c', 'source_b', 'source_d', 'source_e','source_m', 'source_o', 'source_h',
 'source_r',
   'source_s', 'source_t','source_u', 'source_p', 'source_x', 'source_w','Web order','Gender=male', 'Ad
dress_is_res']
df[c] = df[c].astype(str)

#Split Train and Test Data
x_train,x_test,y_train,y_test = train_test_split(df.drop(columns=['Spending']),df['Spending'],test_size
=0.2,random_state=9)

#Standardization
from sklearn.preprocessing import MinMaxScaler
mms = MinMaxScaler()
mms.fit(x_train[['Freq','last_update_days_ago','1st_update_days_ago']])
x_train[['Freq','last_update_days_ago','1st_update_days_ago']]=mms.transform(x_train[['Freq','last_upda
te_days_ago','1st_update_days_ago']])
x_test[['Freq','last_update_days_ago','1st_update_days_ago']]=mms.transform(x_test[['Freq','last_update
_days_ago','1st_update_days_ago']])
```

In [77]:
```
#Elastic Net Regression
from sklearn.linear_model import ElasticNet
grid={'alpha':[1,2,3,4,5,6,7,8,9,10],
      'l1_ratio':[0.3,0.5,0.7]}
EN_linear=ElasticNet(random_state=99)
model = GridSearchCV(estimator=EN_linear, param_grid=grid, cv=5,scoring='neg_mean_squared_error')
model.fit(x_train,y_train)
print("With CV grid search, I found the best hyperparameter is alpha={} and L1 ratio={}.".format(model.
best_params_['alpha'],model.best_params_['l1_ratio']))
print("MSE on Test Data: {}".format(round((metrics.mean_squared_error(y_test, model.predict(x_test))),2
)))
```

```
With CV grid search, I found the best hyperparameter is alpha=1 and L1 ratio=0.7.
MSE on Test Data: 58009.48
```

In [78]:
```
#KNN regression
from sklearn.neighbors import KNeighborsRegressor
grid={'n_neighbors':[5,10,15,20,50,100,500],
      'weights':['uniform','distance']}
KNN=KNeighborsRegressor()
model = GridSearchCV(estimator=KNN, param_grid=grid, cv=5,scoring='neg_mean_squared_error')
model.fit(x_train,y_train)
print("With CV grid search, I found the best hyperparameter is n_neighbors={} and weights={}.".format(m
odel.best_params_['n_neighbors'],model.best_params_['weights']))
print("MSE on Test Data: {}".format(round((metrics.mean_squared_error(y_test, model.predict(x_test))),2
)))
```

```
With CV grid search, I found the best hyperparameter is n_neighbors=5 and weights=distance.
MSE on Test Data: 62928.14
```

In [79]:
```
#Regression Tree
from sklearn.tree import DecisionTreeRegressor
grid={'max_depth':[2,3,4,5,6,7,8,9,10,20],
      'min_samples_split':[2,5,10,15,20]}
Tree=DecisionTreeRegressor(random_state=99)
model = GridSearchCV(estimator=Tree, param_grid=grid, cv=5,scoring='neg_mean_squared_error')
model.fit(x_train,y_train)
print("With CV grid search, I found the best hyperparameter is max_depth={} and min_sample_split={}.".f
ormat(model.best_params_['max_depth'],model.best_params_['min_samples_split']))
print("MSE on Test Data: {}".format(round((metrics.mean_squared_error(y_test, model.predict(x_test))),2
)))
```

```
With CV grid search, I found the best hyperparameter is max_depth=3 and min_sample_split=2.
MSE on Test Data: 38713.35
```

In [80]:
```
#SVM Regression
from sklearn.svm import SVR
grid={'kernel':['linear','rbf'],
      'C':[1,2,3,4,5,6,7,8,9,10]}
s=SVR()
model = GridSearchCV(estimator=s, param_grid=grid, cv=5,scoring='neg_mean_squared_error')
model.fit(x_train,y_train)
print("With CV grid search, I found the best hyperparameter is kernel={} and C={}.".format(model.best_p
arams_['kernel'],model.best_params_['C']))
print("MSE on Test Data: {}".format(round((metrics.mean_squared_error(y_test, model.predict(x_test))),2
)))
```

```
With CV grid search, I found the best hyperparameter is kernel=linear and C=10.
MSE on Test Data: 51984.36
```

In [81]:
```
# Random Search With Neural Network
from tensorflow import keras
from tensorflow.keras import layers
from kerastuner.tuners import RandomSearch
```

In [82]:
```
def build_model(hp):
    model = keras.Sequential()
    model.add(layers.Dense(units=hp.Int('units',
                                        min_value=10,
                                        max_value=100,
                                        step=10),
                           activation='relu',input_dim=22))
    model.add(layers.Dense(units=hp.Int('units',
                                        min_value=10,
                                        max_value=100,
                                        step=10),
                           activation='relu'))
    model.add(layers.Dense(1,activation='linear'))
    model.compile(optimizer=keras.optimizers.Adam(hp.Choice('learning_rate',
                                                            values=[0.01,0.001,0.0001])),
                 loss='mse',
                 metrics=['mse'])
    return model

tuner=RandomSearch(build_model,
                   objective='mse',
                   max_trials=3,
                   overwrite=True,
                   executions_per_trial=3)

#Keras cannot input object data type, so no matter the column is boolean or numeric we need to transfor
m them to float32
x_train_float = np.asarray(x_train).astype(np.float32)
y_train_float = np.asarray(y_train).astype(np.float32)
x_test_float = np.asarray(x_test).astype(np.float32)
y_test_float = np.asarray(y_test).astype(np.float32)

tuner.search(x=x_train_float,y=y_train_float,epochs=200,batch_size=32,validation_data=(x_test_float,y_t
est_float))
```

```
Trial 3 Complete [00h 00m 27s]
mse: 37634.412760416664

Best mse So Far: 9512.133463541666
Total elapsed time: 00h 01m 20s
INFO:tensorflow:Oracle triggered exit
```

In [83]:
```
result=tuner.get_best_hyperparameters()[0].values
print('The best 3 layers NN parameters would be {} neurons and {} learning rate.'.format(result['units'
],result['learning_rate']))
print('-----------------------------------------')
print("The best model's mse on test data = 9512")
print('-----------------------------------------')
print(tuner.results_summary())
```

```
The best 3 layers NN parameters would be 80 neurons and 0.01 learning rate.
-----------------------------------------
The best model's mse on test data = 9512
-----------------------------------------
Results summary
Results in ./untitled_project
Showing 10 best trials
Objective(name='mse', direction='min')
Trial summary
Hyperparameters:
units: 80
learning_rate: 0.01
Score: 9512.133463541666
Trial summary
Hyperparameters:
units: 60
learning_rate: 0.001
Score: 19139.3359375
Trial summary
Hyperparameters:
units: 90
learning_rate: 0.0001
Score: 37634.412760416664
None
```

In [84]:
```
#Random Forest Regression
from sklearn.ensemble import RandomForestRegressor
grid={'n_estimators':[100,200,300,400,500],
      'max_depth':[1,2,3,4,5,6,7,8,9,10]}
rf=RandomForestRegressor()
model = GridSearchCV(estimator=rf, param_grid=grid, cv=5,scoring='neg_mean_squared_error')
model.fit(x_train,y_train)
print("With CV grid search, I found the best hyperparameter is n_estimators={} and max_depth={}.".forma
t(model.best_params_['n_estimators'],model.best_params_['max_depth']))
print("MSE on Test Data: {}".format(round((metrics.mean_squared_error(y_test, model.predict(x_test))),2
)))
```

```
With CV grid search, I found the best hyperparameter is n_estimators=400 and max_depth=4.
MSE on Test Data: 39281.85
```

(c)

For full data, including **both purchased and non-purchased data**, **tree based models (Regression Tree, Ramdom Forest Regression) and Neural Network have the better performances**. On the other hand, for **purchased data only**, **tree based models (Regression Tree, Random Forest Regression) and Neural Network also have the better performances**. In general, **after the purchased data is excluded, all models' performances droppped dramatically**. I believe the reason is **because those who do not purchase and spend 0 dollars have very similar traits. On the other hand, those who do purchase and spend more than 0 dollar have very diverse traits**. Therefore, out models' performances drop dramatically.