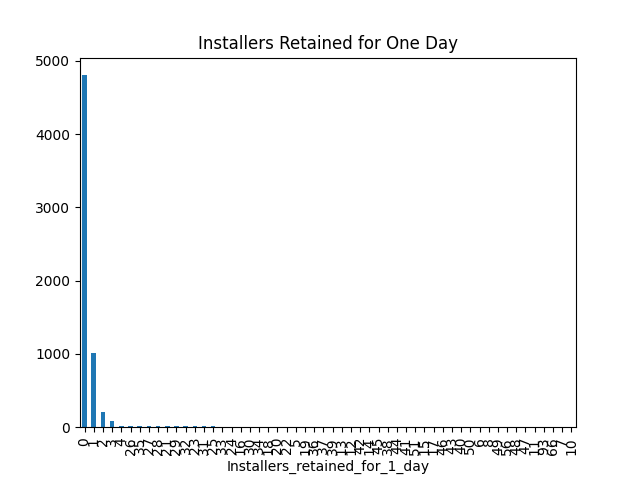
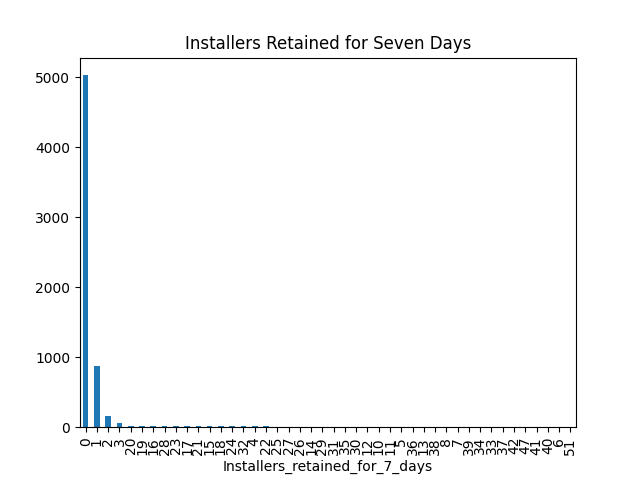
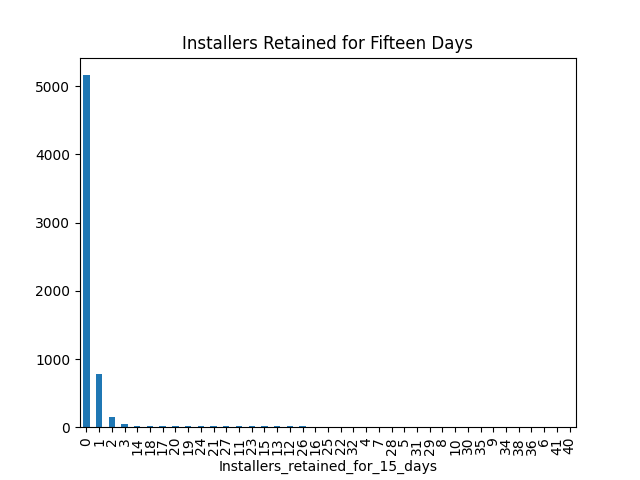
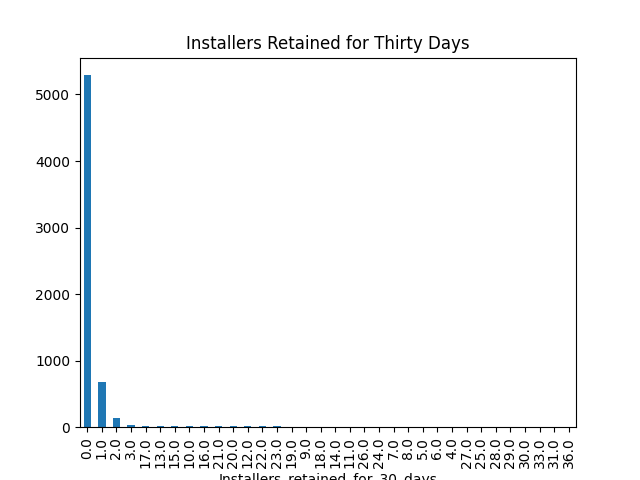
**IN498 Unit 5 Laurence Burden**

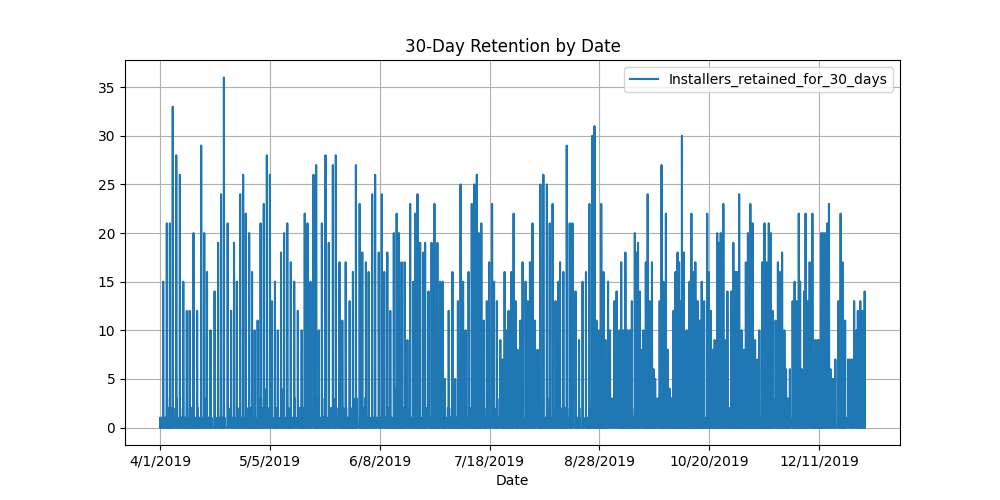
**Plots**

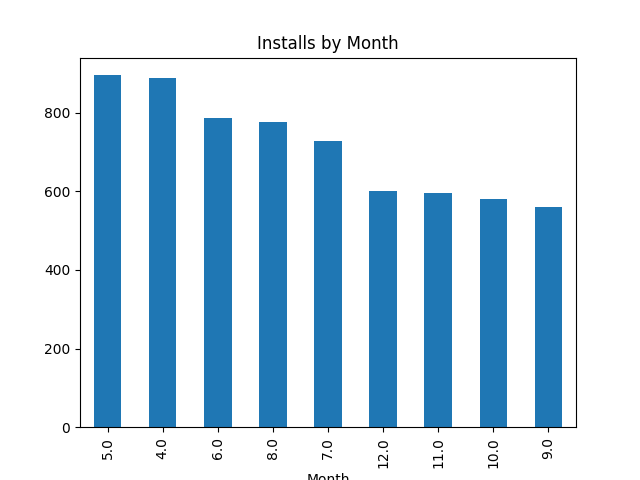
****

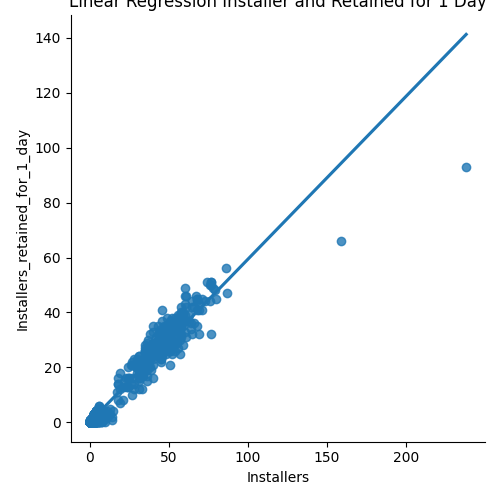
****

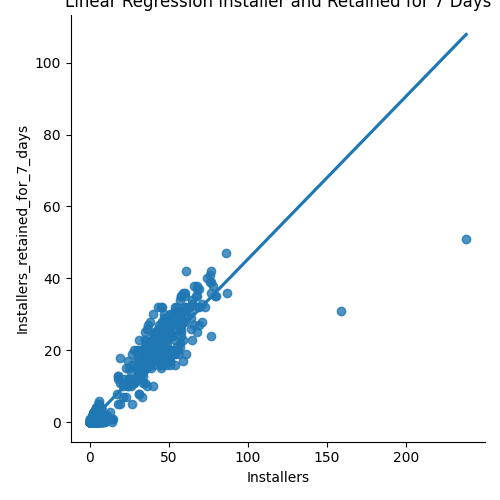
****

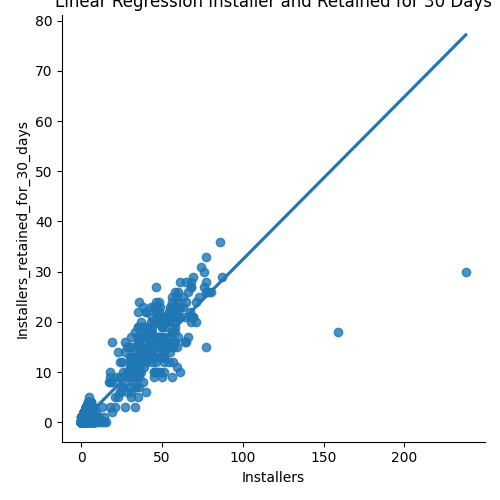
****

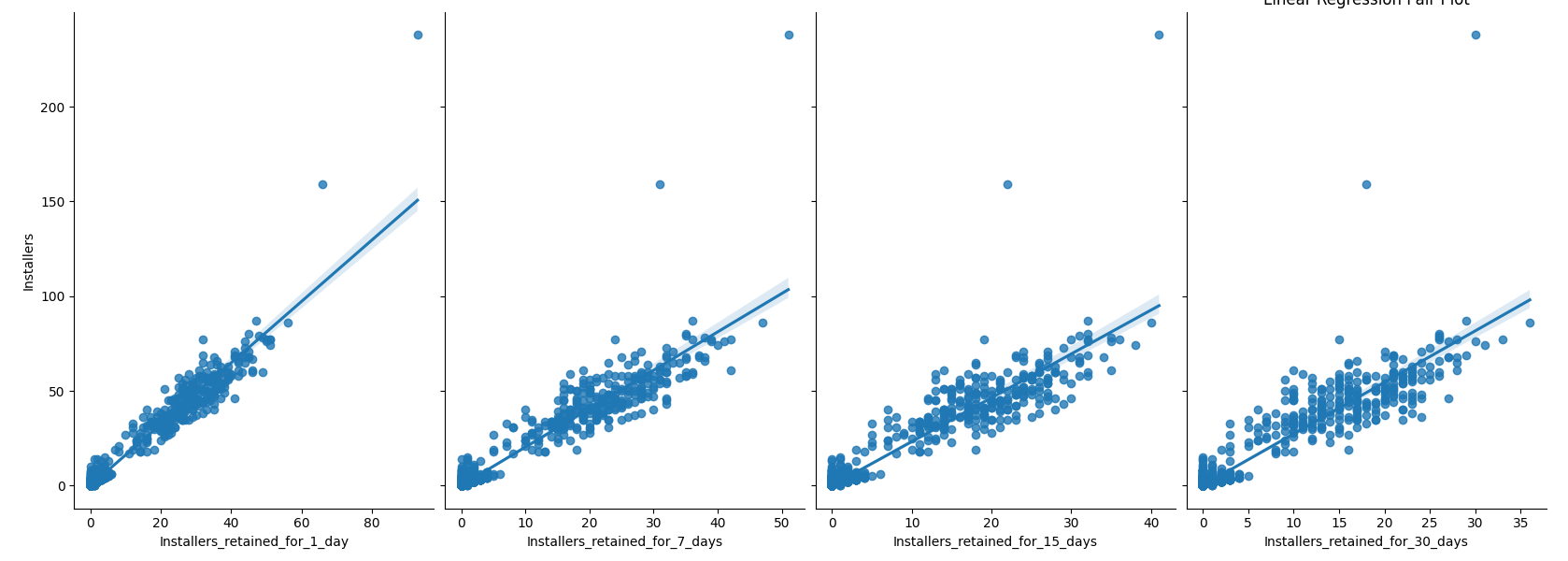
****

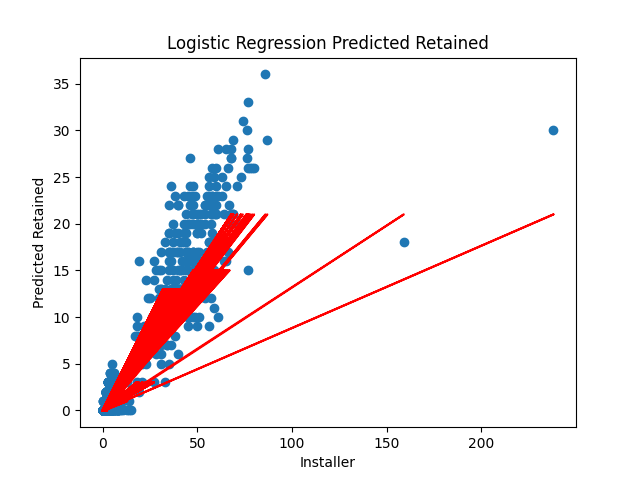
****

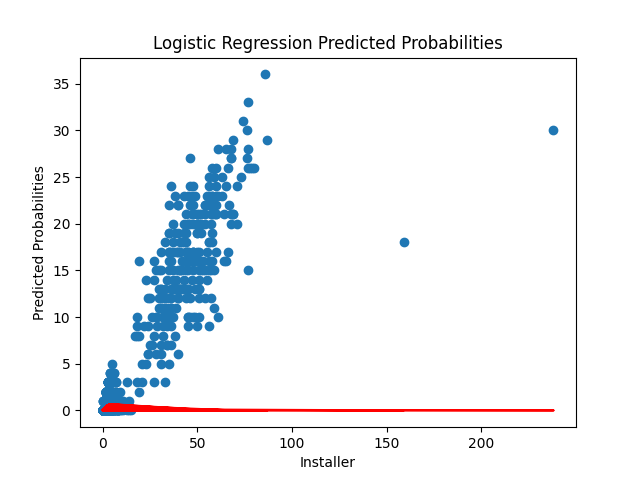
****

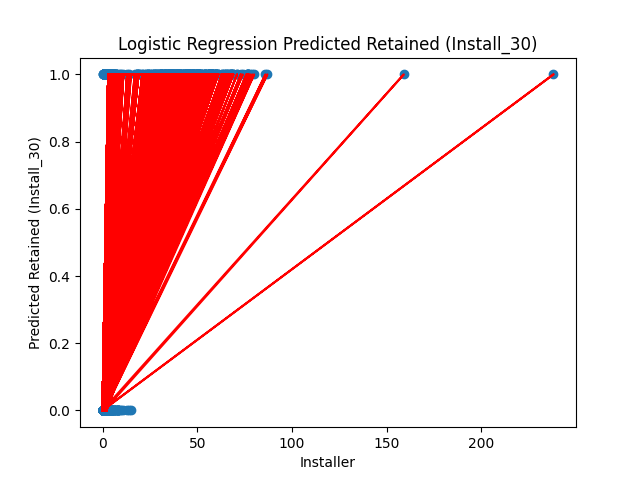
****

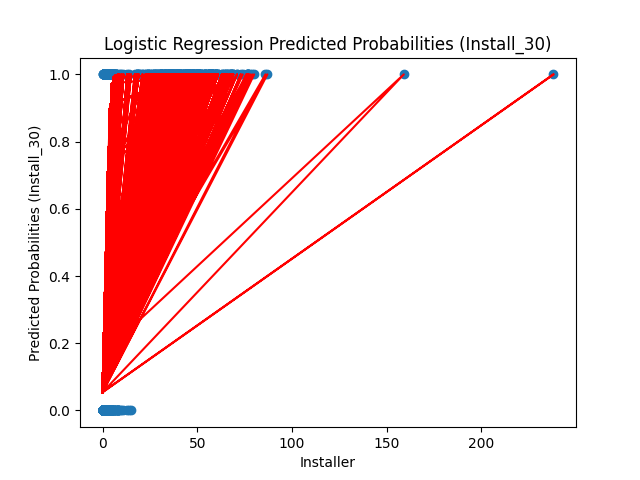
****

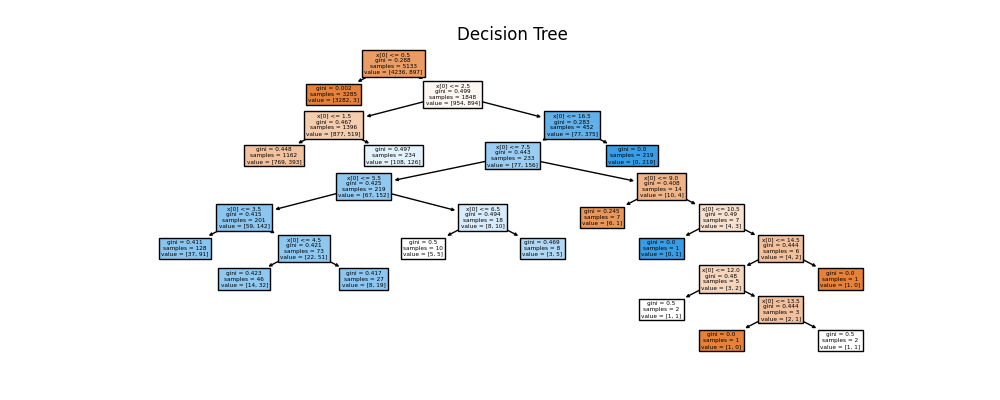
****

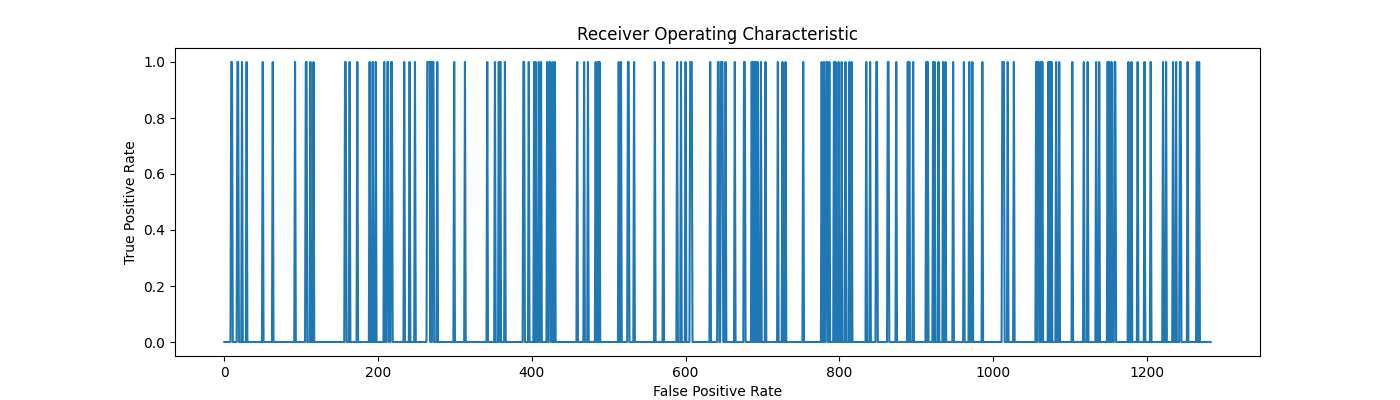
****

****

****

****

****

****

**Code**

import pandas as pd  
import numpy as np  
import seaborn  
from sklearn.linear\_model import LinearRegression  
from sklearn.linear\_model import LogisticRegression  
from sklearn.tree import DecisionTreeClassifier  
from sklearn.ensemble import RandomForestClassifier  
from sklearn.model\_selection import train\_test\_split  
from sklearn.metrics import roc\_auc\_score  
from sklearn import metrics  
import sys  
import seaborn as sns  
import matplotlib.pyplot as plt  
from sklearn import tree  
  
# Ignoring warnings  
if not sys.warnoptions:  
 import warnings  
  
 warnings.simplefilter("ignore")  
  
# Widen the column display  
pd.set\_option('max\_colwidth', 500)  
  
# Read data into a DataFrame using these columns  
# "Date","Package\_Name","Country","Store\_Listing\_Visitors",  
# "Installers","Visitor-to-Installer\_conversion\_rate",  
# "Installers\_retained\_for\_1\_day","Installer-to-1\_day\_retention\_rate",  
# "Installers\_retained\_for\_7\_days","Installer-to-7\_days\_retention\_rate",  
# "Installers\_retained\_for\_15\_days","Installer-to-15\_days\_retention\_rate",  
# "Installers\_retained\_for\_30\_days","Installer-to-30\_days\_retention\_rate"  
col\_names = ["Date", "Package\_Name", "Country", "Store\_Listing\_Visitors", "Installers",  
 "Visitor-to-Installer\_conversion\_rate", "Installers\_retained\_for\_1\_day",  
 "Installer-to-1\_day\_retention\_rate", "Installers\_retained\_for\_7\_days",  
 "Installer-to-7\_days\_retention\_rate", "Installers\_retained\_for\_15\_days",  
 "Installer-to-15\_days\_retention\_rate", "Installers\_retained\_for\_30\_days",  
 "Installer-to-30\_days\_retention\_rate"]  
data = pd.read\_csv('retained\_installers\_com.foo.bar\_Combined.csv', names=col\_names)  
  
########### EXPLORE THE ORIGINAL DATA SET #######################  
# sns.barplot(data['Installers\_retained\_for\_1\_day'].value\_counts())  
data['Installers\_retained\_for\_1\_day'].value\_counts().plot(title='Installers Retained for One Day',  
 kind='bar')  
plt.show()  
  
# Plot the installer retention rates for 7 days using a bar chart  
data['Installers\_retained\_for\_7\_days'].value\_counts().plot(title='Installers Retained for Seven Days',  
 kind='bar')  
plt.show()  
  
# Plot the installer retention rates for 15 days using a bar chart  
data['Installers\_retained\_for\_15\_days'].value\_counts().plot(title='Installers Retained for Fifteen Days',  
 kind='bar')  
plt.show()  
  
# Plot the installer retention rates for 30 days using a bar chart  
data['Installers\_retained\_for\_30\_days'].value\_counts().plot(title='Installers Retained for Thirty Days',  
 kind='bar')  
plt.show()  
  
# Plot the retention of users at 30 days per date using a line chart  
# data['Installers\_retained\_for\_30\_days'].value\_counts().plot(title='Installers Retained for Thirty Days',  
# kind='line')  
data.plot(x='Date', y='Installers\_retained\_for\_30\_days', figsize=(10, 5), grid=True)  
plt.title('30-Day Retention by Date')  
plt.show()  
  
# Add a month column by converting the Date column for month only  
data['Month'] = pd.DatetimeIndex(data['Date']).month  
  
# Plot the installs per month using a bar chart  
data['Month'].value\_counts().plot(title='Installs by Month', kind='bar')  
plt.show()  
  
########### EARLY SEABORN PLOTS #######################  
# Seaborn simple regression plot of installers retained for 1 day  
sns.lmplot(x='Installers', y='Installers\_retained\_for\_1\_day', data=data, ci=None)  
plt.title('Linear Regression Installer and Retained for 1 Day')  
plt.show()  
  
# Seaborn simple regression plot of installers retained for 7 day  
sns.lmplot(x='Installers', y='Installers\_retained\_for\_7\_days', data=data, ci=None)  
plt.title('Linear Regression Installer and Retained for 7 Days')  
plt.show()  
  
# Seaborn simple regression plot of installers retained for 30 day  
sns.lmplot(x='Installers', y='Installers\_retained\_for\_30\_days', data=data, ci=None)  
plt.title('Linear Regression Installer and Retained for 30 Days')  
plt.show()  
  
############ FIX MISSING DATA #######################  
# Replace NaN with 0 for Installers\_retained\_for\_30\_days  
data[np.isnan(data.Installers\_retained\_for\_30\_days)] = 0  
  
############ SEABORN PLOT AFTER FIXING MISSING DATA #######################  
# Seaborn pairplot for installers retained for 1, 7, 15, and 30 days  
sns.pairplot(data,  
 x\_vars=['Installers\_retained\_for\_1\_day', 'Installers\_retained\_for\_7\_days',  
 'Installers\_retained\_for\_15\_days', 'Installers\_retained\_for\_30\_days'],  
 y\_vars='Installers', size=6, aspect=0.7, kind='reg')  
plt.title('Linear Regression Pair Plot')  
plt.show()  
  
########### LOGISTIC REGRESSION #######################  
# Get logistic regression model  
logreg = LogisticRegression()  
  
# Set feature\_cols to Installers column data only  
feature\_cols = ['Installers']  
  
# Set X to feature\_cols  
X = data[feature\_cols]  
  
# Set y to installer retained for 30 days  
y = data.Installers\_retained\_for\_30\_days  
  
# Fit X and y to logistic regression model  
logreg.fit(X, y)  
  
# Predict classes using X and save as a variable called assorted\_pred\_class  
assorted\_pred\_class = logreg.predict(X)  
  
# Plot the predictions in a scatter plot using  
# Scatter uses data.Installers and data.Installers\_retained\_for\_30\_days  
# X label of Installer  
# y label of predicted retained  
# Plot using data.Installer and the prediction results (assorted\_pred\_class) on X  
plt.scatter(data.Installers, data.Installers\_retained\_for\_30\_days)  
plt.plot(data.Installers, assorted\_pred\_class, color='red')  
plt.xlabel('Installer')  
plt.ylabel('Predicted Retained')  
plt.title('Logistic Regression Predicted Retained')  
plt.show()  
  
# Get the predicted probabilites of class 1  
# Save as a variable called assorted\_pred\_prob  
assorted\_pred\_prob = logreg.predict\_proba(X)[:, 1]  
  
# Scatter plot the predicted probabilities  
# Scatter uses data.Installers and data.Installers\_retained\_for\_30\_days  
# X label of Installer  
# y label of predicted probabilities  
# Plot using data.Installer and the probability results (assorted\_pred\_prob) on X  
plt.scatter(data.Installers, data.Installers\_retained\_for\_30\_days)  
plt.plot(data.Installers, assorted\_pred\_prob, color='red')  
plt.xlabel('Installer')  
plt.ylabel('Predicted Probabilities')  
plt.title('Logistic Regression Predicted Probabilities')  
plt.show()  
  
#################################### INSTALL\_30 ##########################################  
# Add a new column for installers retained for 30 days  
# Call the new column Install\_30.  
# If greater than 0, put 1, if 0, put 0  
data['Install\_30'] = np.where(data['Installers\_retained\_for\_30\_days'] > 0, 1, 0)  
  
########### LOGISTIC REGRESSION #######################  
  
# Perform logistic regression using Install\_30 column for y  
# X = Installers column  
logreg = LogisticRegression()  
feature\_cols = ['Installers']  
X = data[feature\_cols]  
y = data.Install\_30  
  
# Fit X and y for logistic regression  
logreg.fit(X, y)  
  
# Predict on X and capture the result to assorted\_pred\_class  
assorted\_pred\_class = logreg.predict(X)  
  
# Plot the class predictions  
# Scatter uses data.Intsallers and data.Install\_30  
# Plot uses data.Installers and assorted\_pred\_class  
# X label is Installer  
# Y label is Predicted Retained (Install\_30)  
plt.scatter(data.Installers, data.Install\_30)  
plt.plot(data.Installers, assorted\_pred\_class, color='red')  
plt.xlabel('Installer')  
plt.ylabel('Predicted Retained (Install\_30)')  
plt.title('Logistic Regression Predicted Retained (Install\_30)')  
plt.show()  
  
# Get the predicted probabilites of class 1 and save to assorted\_pred\_prob  
assorted\_pred\_prob = logreg.predict\_proba(X)[:, 1]  
  
# Plot the predicted probabilities  
# Scatter uses data.Intsallers and data.Install\_30  
# Plot uses data.Installers and assorted\_pred\_prob  
# X label is Installer  
# Y label is Predicted Probabilities (Install\_30)  
plt.scatter(data.Installers, data.Install\_30)  
plt.plot(data.Installers, assorted\_pred\_prob, color='red')  
plt.xlabel('Installer')  
plt.ylabel('Predicted Probabilities (Install\_30)')  
plt.title('Logistic Regression Predicted Probabilities (Install\_30)')  
plt.show()  
  
########### DECISION TREE #######################  
# Train/test split 80% train, 20% test  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2)  
  
# Build a decision tree model  
dt = DecisionTreeClassifier(random\_state=1)  
  
# Fit the tree using X\_train and y\_train  
dt.fit(X\_train, y\_train)  
  
# Plot the decision tree  
plt.figure(figsize=(10,4))  
tree.plot\_tree(dt, filled=True)  
plt.title('Decision Tree')  
plt.show()  
  
print(tree.export\_text(dt))  
  
########### RANDOM FOREST #######################  
# Build a random forest tree model  
# n\_estimators = 1000  
rf = RandomForestClassifier(n\_estimators=1000)  
  
# Fit the random tree model with X\_train and y\_train  
rf.fit(X\_train, y\_train)  
  
# Get the predicitons using X\_test and save to rf\_probs  
rf\_probs = rf.predict(X\_test)  
  
# Plot the random forest predictions using rf\_probs  
plt.figure(figsize=(14,4))  
plt.plot(rf\_probs)  
plt.xlabel('False Positive Rate')  
plt.ylabel('True Positive Rate')  
plt.title('Receiver Operating Characteristic')  
plt.show()