```
#%%
import numpy as np
import pandas as pd
import sns as sns

#%% md
1. Read the Auto data (5 points)
    a. use pandas to read the data
    b. output the first few rows
    c. output the dimensions of the data
#%%
data = pd.read_csv("Auto.csv")
print(data)
#%%
print(data.head())
print("Dimensions:", data.shape)
#%% md
2. Data exploration with code (5 points)
    a. use describe() on the mpg, weight, and year columns
    b. write comments indicating the range and average of each column
#%%
print("MPG column:")
print(data["mpg"].describe())

print("Weight column:")
print(data["weight"].describe())

print("Year column:")
print(data["year"].describe())
#%% md
MPG Column: Range = 37.6 , Mean = 23.45
Weight Column: Range = 3527 , Mean = 2977.58
Year Column: Range = 12 , Mean = 76.01
#%% md
3. Explore data types (5 points)
    a. check the data types of all columns
    b. change the cylinders column to categorical ( use cat.codes)
    c. change the origin column to categorical (don
't use cat.codes)
    d. verify the changes with the dtypes attribute
#%%
print(data.dtypes)
#%%
data["cylinders"] = data["cylinders"].astype("
category")
data["cylinders_cat"] = data["cylinders"].cat.codes

data["origin"] = data["origin"].astype("category")

print(data.dtypes)
print(data.shape)
#%% md
4. Deal with NAs (5 points)
    a. delete rows with NAs
    b. output the new dimensions
#%%
print(data.shape)
data.dropna(inplace=True)
print("Dimensions after deleting rows with NAs:")
print(data.shape)
#%% md
5. Modify columns (10 points)
    a. make a new column, mpg_high, and make it
categorical:
        i. the column == 1 if mpg > average mpg,
else == 0
    b. delete the mpg and name columns (delete mpg
so the algorithm doesn't just learn
    to predict mpg_high from mpg)
    c. output the first few rows of the modified
data frame
#%%
avg_mpg = data['mpg'].mean()
data['mpg_high'] = (data['mpg'] > avg_mpg).astype(
int)

data.drop(columns=['mpg', 'name'], inplace=True)

print(data.head())
```

```
#%% md
6. Data exploration with graphs (15 points)
    a. seaborn catplot on the mpg_high column
    b. seaborn relplot with horsepower on the x axis, weight on the y axis, setting hue or
    style to mpg_high
    c. seaborn boxplot with mpg_high on the x axis and weight on the y axis
    d. for each graph, write a comment indicating one thing you learned about the data
    from the graph
#%%
import seaborn as sns
import matplotlib.pyplot as plt

sns.catplot(x="mpg_high", kind="count", data=data)
sns.relplot(x="horsepower", y="weight", hue="mpg_high", data=data)
sns.boxplot(x="mpg_high", y="weight", data=data)
plt.show()
#%% md
Catplot: We can see that there are more vehicles that have mpg lower than the average. However the numbers are really close.
Retplot and Boxplot: we can see that the cars with a lower weight has less mpg. This makes sense because the engine now requires less amount of work to move the vehicle.
#%% md
7. Train/test split (5 points)
    a. 80/20
    b. use seed 1234 so we all get the same results
    c. train /test X data frames consists of all remaining columns except mpg_high
    d. output the dimensions of train and test
#%%
from sklearn.model_selection import train_test_split

np.random.seed(1234)
```

```
X = data.drop('mpg_high', axis=1)
y = data['mpg_high']

X_train, X_test, y_train, y_test =
train_test_split(X, y, test_size=0.2)

print("Train:", X_train.shape, y_train.shape)
print("Test:", X_test.shape, y_test.shape)
#%% md
8. Logistic Regression (10 points)
    a. train a logistic regression model using
solver lbfgs
    b. test and evaluate
    c. print metrics using the classification
report
#%%
from sklearn.linear_model import
LogisticRegression
from sklearn.metrics import classification_report

clasf = LogisticRegression(solver='lbfgs')
clasf.fit(X_train, y_train)

y_pred = clasf.predict(X_test)
print(classification_report(y_test, y_pred))
#%% md
9. Decision Tree (10 points)
    a. train a decision tree
    b. test and evaluate
    c. print the classification report metrics
#%%
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report

tree = DecisionTreeClassifier(max_depth=3,
random_state=1234)
tree.fit(X_train, y_train)
y_pred = tree.predict(X_test)

print(classification_report(y_test, y_pred))
#%% md
```

```
10. Neural Network (15 points)
    a. train a neural network, choosing a network topology of your choice
    b. test and evaluate
    c. train a second network with a different topology and different settings
    d. test and evaluate
    e. compare the two models and why you think the performance was same/different
#%%
from sklearn.neural_network import MLPClassifier

nn = MLPClassifier(hidden_layer_sizes=(20,), max_iter=700, random_state=1234)
nn.fit(X_train, y_train)
y_pred = nn.predict(X_test)
print("Neural Network 1:")
print(classification_report(y_test, y_pred))

nn2 = MLPClassifier(hidden_layer_sizes=(20, 10), max_iter=1000, random_state=1234)
nn2.fit(X_train, y_train)
y_pred2 = nn2.predict(X_test)
print("Neural Network 2:")
print(classification_report(y_test, y_pred2))
#%% md
The two neural networks have very similar results . The First one is predicting better results for class 1 while the second one is producing better results for class 0. The result are same because the topology being very similar too.
#%% md
11. Analysis (15 points)
    a. which algorithm performed better?
    It looks like the logistic algorithm performed the best.
    b. compare accuracy, recall and precision metrics by class
    By comparing all three we can still see that logistic expression is giving us the best results.
    c. give your analysis of why the better-
```

```
performing algorithm might have outperformed
    the other
    We can see that Logistic expression is giving us the best accuracy. This is because simple algorithm like logistic expression is suited for this problem
    d. write a couple of sentences comparing your experiences using R versus sklearn. Feel
    free to express strong preferences.
    Python is definitely easier for me because I did a similar but more complexive project like this last semester which is why I had some previous experience. However in R, it is my first time trying it so it was quite difficult for me at the beginning.
```