P1 (50pt): Write a Python code in Colab using NumPy, Panda, Scikit-Learn to complete the following tasks:

```
import pandas as pd
import numpy as np
np.random.seed(100)
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split, cross_val_score, GridSearchCV
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.ensemble import RandomForestClassifier
```

1. Import the Auto MPG dataset with pandas.read_csv() using the dataset URL, use the attribute names as explained in the dataset description as the column names **(5pt)**, view the strings '?' as the missing value, and whitespace (i.e., '\s+') as the column delimiter. Print out the shape and first 5 rows of the obtained DataFrame. **(5pt)**

- Dataset source file: http://archive.ics.uci.edu/ml/machine-learning-databases/auto-mpg/auto-mpg.data
- Dataset description: http://archive.ics.uci.edu/ml/datasets/Auto+MPG
- To have reproducible results using Scikit-Learn in Jupyter Notebook for the training/testing, set the seed at the beginning of your notebook https://www.mikulskibartosz.name/how-to-set-the-global-random_state-in-scikit-learn/

```
url= "http://archive.ics.uci.edu/ml/machine-learning-databases/auto-mpg/auto-mpg.data"
colnames = ['mpg', "cylinders", 'displacement', 'horsepower', 'weight','acceleration','model year', 'origin','car n
df = pd.DataFrame(pd.read_csv(url, names= colnames, delimiter ='\s+',na_values= '?'))
df.head()
```

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	or
0	18.0	8	307.0	130.0	3504.0	12.0	70	
1	15.0	8	350.0	165.0	3693.0	11.5	70	

```
df.shape
```

```
(398, 9)
```

2. Delete the "car_name" column using .drop() method as it is irrelevant to the prediction. Print out a concise summary of the new DataFrame using .info() and check if NULL value exists in each column **(5pt)**

```
df.drop('car name', axis = 1, inplace= True)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 398 entries, 0 to 397
Data columns (total 8 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   mpg           398 non-null    float64
 1   cylinders     398 non-null    int64
 2   displacement  398 non-null    float64
 3   horsepower    392 non-null    float64
```

```
 4   weight         398 non-null    float64
 5   acceleration   398 non-null    float64
 6   model year     398 non-null    int64
 7   origin         398 non-null    int64
dtypes: float64(5), int64(3)
memory usage: 25.0 KB
```

3. Replace the NULL value with the mean value of the column using .fillna(). Print out the concise summary of the new DataFrame and recheck if NULL value exists in each column **(5pt)**

```
mean_value = df['horsepower'].mean()
df['horsepower'].fillna(value= mean_value, inplace = True)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 398 entries, 0 to 397
Data columns (total 8 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   mpg            398 non-null    float64
 1   cylinders      398 non-null    int64
 2   displacement   398 non-null    float64
 3   horsepower     398 non-null    float64
 4   weight         398 non-null    float64
 5   acceleration   398 non-null    float64
 6   model year     398 non-null    int64
 7   origin         398 non-null    int64
dtypes: float64(5), int64(3)
memory usage: 25.0 KB
```

4. For the 'origin' column with categorical attribute, replace it with the columns with numerical attributes using one-hot encoding (you can use either get_dummies() in Pandas or OneHotEncoder in Scikit-Learn). Print out the first 5 rows of the newly obtained DataFrame. **(10pt)**

   a. https://stackabuse.com/one-hot-encoding-in-python-with-pandas-and-scikit-learn/

```
pd.get_dummies(df['origin'])
```

	1	2	3
0	1	0	0
1	1	0	0
2	1	0	0
3	1	0	0
4	1	0	0
...	...	...	...
393	1	0	0
394	0	1	0
395	1	0	0
396	1	0	0
397	1	0	0

398 rows × 3 columns

```
df.head()
```

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	or

0	18.0	8	307.0	130.0	3504.0	12.0	70
1	15.0	8	350.0	165.0	3693.0	11.5	70
2	18.0	8	318.0	150.0	3436.0	11.0	70
3	16.0	8	304.0	150.0	3433.0	12.0	70
4	17.0	8	302.0	140.0	3449.0	10.5	70

5. Learn a linear regression model (fit_intercept=True) to predict the "mpg" column from the remaining columns in the obtained DataFrame of Step 4.

   a. Separate the "mpg" column from other columns and view it as the label vector and others as the feature matrix **(5pt)**

   b. Split the data into a training set (80%) and testing set (20%) using train_test_split and print out their shapes **(5pt)**

```
y = df['mpg']
X = df.drop('mpg', axis =1)
X.shape
X1, X2, y1, y2 = train_test_split(X, y, test_size=0.2)
```

c. Train the model using the training set and print out the coefficients of the model **(5 pt)**

```
reg =LinearRegression(fit_intercept=True)
reg.fit(X1, y1)
reg.coef_
```

```
array([-0.31833549,  0.01730294, -0.00959021, -0.00680533,  0.0941134 ,
        0.78561411,  1.34089894])
```

d. Use the learned model to predict on the test set and print out the mean squared error of the predictions **(5pt)**

```
ypred = reg.predict(X2)
mean_squared_error(ypred,y2)
```

```
9.157651635045296
```

P2 (50pt): Write a Python code in Colab using NumPy, Panda, Scikit-Learn to complete the following tasks:

1. Import the red wine dataset with pandas.read_csv() using the dataset URL, use the semi-colon as the column delimiter, and print out both the first five rows and a concise summary of the obtained DataFrame. **(10 pt)**

   a. Dataset source file: http://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-red.csv

   b. Dataset description: http://archive.ics.uci.edu/ml/datasets/wine+quality

   c. To have reproducible results using Scikit-Learn in Jupyter Notebook for the training/testing, set the seed at the beginning of your notebook https://www.mikulskibartosz.name/how-to-set-the-global-random_state-in-scikit-learn/

```
url= "http://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-red.csv"
df = pd.DataFrame(pd.read_csv(url, delimiter =';'))
df.head()
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density

0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1599 entries, 0 to 1598
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   fixed acidity         1599 non-null   float64
 1   volatile acidity      1599 non-null   float64
 2   citric acid           1599 non-null   float64
 3   residual sugar        1599 non-null   float64
 4   chlorides             1599 non-null   float64
 5   free sulfur dioxide   1599 non-null   float64
 6   total sulfur dioxide  1599 non-null   float64
 7   density               1599 non-null   float64
 8   pH                    1599 non-null   float64
 9   sulphates             1599 non-null   float64
 10  alcohol               1599 non-null   float64
 11  quality               1599 non-null   int64  
dtypes: float64(11), int64(1)
memory usage: 150.0 KB
```

2. Suppose we want to predict the quality of wine from other attributes. Divide the data into a label vector and a feature matrix. Then split them into a training set (80%) and testing set (20%) using train_test_split. **(5pt)**

```
y = df['quality']
X = df.drop('quality', axis =1)
X1, X2, y1, y2 = train_test_split(X, y, test_size=0.2)
```

3. Use the StandardScalar() in Scikit-Learn to preprocess the feature matrices of both training set and testing set. Note that the testing set can only be scaled by the mean and standard deviation values obtained from the training set. **(5 pt)**

   a. https://scikit-learn.org/stable/modules/preprocessing.html (Refer to the code examples when using MinMaxScalar() in Section 6.3.1.1)

```
from sklearn import preprocessing
scaler = preprocessing.StandardScaler().fit(X1)
X1_scaled = scaler.transform(X1) #scaled training dataset
X1_scaled.mean(axis=0) # check mean = 0
X1_scaled.std(axis=0) # check std = 1
X2_scaled = scaler.transform(X2)
```

4. Use the preprocessed training dataset to learn a classification model with RandomForestClassifier (with 300 trees) in Scikit-Learn. Use 5-fold cross-validation to train and cross-validate the model on the preprocessed training dataset. Print out the accuracies returned by the five folds as well as their average and standard deviation values. **(10 pt)**

```
rf_clf = RandomForestClassifier(n_estimators= 300)
cv_score = cross_val_score(rf_clf, X1_scaled, y1, cv = 5)
```

```
cv_score.mean()
```

```
0.6700490196078431
```

```
cv_score.std()
```

```
0.012881176506638102
```

5. Use GridSearchCV() to find and print out the best hyperparameter for the number of trees (try the following values: 100, 300, 500, 800, 1000 for 'n_estimator'), ignoring the search for other hyperparameters. Also use 5-fold cross-validation during GridSearchCV. **(15 pt)**

```
from sklearn.model_selection import GridSearchCV
params = {'n_estimators': [100,300,500,800,1000]}
grid = GridSearchCV(rf_clf, params, cv = 5)
grid.fit(X1_scaled,y1)
```

```
GridSearchCV(cv=5, estimator=RandomForestClassifier(n_estimators=300),
             param_grid={'n_estimators': [100, 300, 500, 800, 1000]})
```

```
grid.best_params_
```

```
{'n_estimators': 300}
```

6. Find and print out the testing accuracy of the model obtained using the best hyperparameter value on the preprocessed testing dataset **(5 pt)**

```
from sklearn.metrics import accuracy_score
ypred =grid.predict(X2_scaled)
accuracy_score(y2,ypred)
```

```
0.684375
```