APPLIED DATA SCIENCE

PHASE-3 SUBMISSION

STOCK PRICE PREDICTION

For machine learning algorithms to work, it’s necessary to convert **raw data** into a **clean data** set, which means we must convert the data set to **numeric data**. We do this by encoding all the **categorical labels** to column vectors with binary values. **Missing values**, or NaNs (not a number) in the data set is an annoying problem. You have to either drop the missing rows or fill them up with a mean or interpolated values.

Preprocess data in Python – Step by step:

1. Load data in Pandas.
2. Drop columns that aren’t useful.
3. Drop rows with missing values.
4. Create dummy variables.
5. Take care of missing data.
6. Convert the data frame to NumPy.
7. Divide the data set into training data and test data.

1.Load data in Pandas:

To work on the data, you can either load the CSV in Excel or in Pandas. For the purposes of this tutorial, we’ll load the CSV data in Pandas.

```
import pandas as pd
df = pd.read_csv('MSFT.csv')
```

Let's take a look at the data format below:

```
[3] df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8525 entries, 0 to 8524
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Date       8525 non-null   object
 1   Open       8525 non-null   float64
 2   High       8525 non-null   float64
 3   Low        8525 non-null   float64
 4   Close      8525 non-null   float64
 5   Adj Close  8525 non-null   float64
 6   Volume     8525 non-null   int64
dtypes: float64(5), int64(1), object(1)
memory usage: 466.3+ KB
```

2. Drop Columns That Aren't Useful: Let's try to drop some of the columns which won't contribute much to our machine learning model. We'll start with Date and Open.

```
[4] cols = ['Date','Open']
    df = df.drop(cols, axis=1)

df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8525 entries, 0 to 8524
Data columns (total 5 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   High       8525 non-null   float64
 1   Low        8525 non-null   float64
 2   Close      8525 non-null   float64
 3   Adj Close  8525 non-null   float64
 4   Volume     8525 non-null   int64
dtypes: float64(4), int64(1)
memory usage: 333.1 KB
```

3. Drop Rows With Missing Values: Next we can drop all rows in the data that have missing values (NaNs). Here's how:

```
[6] df = df.dropna()
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8525 entries, 0 to 8524
Data columns (total 5 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   High       8525 non-null   float64
 1   Low        8525 non-null   float64
 2   Close      8525 non-null   float64
 3   Adj Close  8525 non-null   float64
 4   Volume     8525 non-null   int64
dtypes: float64(4), int64(1)
memory usage: 333.1 KB
```

4. Creating Dummy Variables

Instead of wasting our data, let's convert the Pclass, Sex and Embarked to columns in Pandas and drop them after conversion.

```
[8] dummies = []
    cols = ['High', 'Low']
    for col in cols:
       dummies.append(pd.get_dummies(df[col]))
```

Then..

```
[9] MSFT_dummies = pd.concat(dummies, axis=1)
```

Finally we **concatenate** to the original data frame, column-wise:

```
[10] df = pd.concat((df,MSFT_dummies), axis=1)
```

Now that we converted High and Low values into columns, we drop the redundant columns from the data frame.

```
[11] df = df.drop(['High','Low'], axis=1)
```

Let's take a look at the new data frame:

```
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8525 entries, 0 to 8524
Columns: 9265 entries, Close to 158.330002
dtypes: float64(2), int64(1), uint8(9262)
memory usage: 75.5 MB
```

5. Take Care of Missing Data

Let’s compute a median or interpolate() all the ages and fill those missing age values. Pandas has an interpolate() function that will replace all the missing NaNs to interpolated values.

```
[13] df['Close'] = df['Close'].interpolate()
```

Now let's observe the data columns. Notice ‘Close’ is now interpolated with imputed new values.

```
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8525 entries, 0 to 8524
Columns: 9265 entries, Close to 158.330002
dtypes: float64(2), int64(1), uint8(9262)
memory usage: 75.5 MB
```

6. Convert the Data Frame to NumPy: Now that we've converted all the data to integers, it's time to prepare the data for machine learning models. This is where scikit-learn and NumPy come into play:

X= Input set with 14 attributes

y = Small y output, in this case Survived

Now we convert our data frame from Pandas to NumPy and we assign input and output:

```
[15] X = df.values
     y = df['Adj Close'].values
```

X still has Adj Close values in it, which should not be there. So we drop in the NumPy column, which is the first column.

```
X = np.delete(X, 1, axis=1)
```

7. Divide the Data Set Into Training Data and Test Data

Now that we're ready with X and y, let's split the data set: we'll allocate 70 percent for training and 30 percent for tests using scikit model_selection.

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=(
```