```
Name: Esplanada, Borris A.
Section: CPE32S1
Instructor: Engr. Roman Richard
Date: 7/01/2024
```

1. Choose any dataset applicable to the classification problem, and also, choose any dataset applicable to the regression problem.
2. Explain your datasets and the problem being addressed.
3. For classification, do the following:
    - Create a base model
    - Evaluate the model with k-fold cross validation
    - Improve the accuracy of your model by applying additional hidden layers
4. For regression, do the following:
    - Create a base model
    - Improve the model by standardizing the dataset
    - Show tuning of layers and neurons (see evaluating small and larger networks)
5. Submit the link to your Google Colab (make sure that it is accessible to me)

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

Classification Problem

Title: Personal Loan Modeling

Link: https://www.kaggle.com/datasets/teertha/personal-loan-modeling

Explain your datasets and the problem being addressed.

The Personal Loan Modeling dataset contains information about customers of a bank, including their personal and financial details, as well as whether or not they accepted a personal loan offer in the past. The dataset consists of 5,000 rows and 14 columns.

The problem being addressed is whether or not a customer will accept a personal loan offer. This is a binary classification problem, where the goal is to predict whether a customer will accept the loan offer or not based on their personal and financial information.

This is important to banks and other financial institutions because they want to make targeted marketing efforts towards customers who are most likely to accept personal loan offers, so they can increase their chances of making a profit.

By using this dataset, banks can gain insights into what factors may influence a customer's decision to accept a personal loan offer and adjust their marketing strategy accordingly.

For classification, do the following:

- Create a base model
- Evaluate the model with k-fold cross validation
- Improve the accuracy of your model by applying additional hidden layers

```
path = "/content/drive/MyDrive/3rdYear/CPE019/hoa7.1/Bank_Personal_Loan_Modelling.csv"
df = pd.read_csv(path)
```

```
pip install scikeras[tensorflow]
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: scikeras[tensorflow] in /usr/local/lib/python3.9/dist-packages (0.10.0)
Requirement already satisfied: scikit-learn>=1.0.0 in /usr/local/lib/python3.9/dist-packages (from scikeras[tensorflow]) (1.2.2)
Requirement already satisfied: packaging>=0.21 in /usr/local/lib/python3.9/dist-packages (from scikeras[tensorflow]) (23.0)
Requirement already satisfied: tensorflow>=2.11.0 in /usr/local/lib/python3.9/dist-packages (from scikeras[tensorflow]) (2.12.0)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.9/dist-packages (from scikit-learn>=1.0.0->scikeras[tensorflow]) (1.2.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.9/dist-packages (from scikit-learn>=1.0.0->scikeras[tensorflow]) (3.1.0)
Requirement already satisfied: numpy>=1.17.3 in /usr/local/lib/python3.9/dist-packages (from scikit-learn>=1.0.0->scikeras[tensorflow]) (1.22.4)
Requirement already satisfied: scipy>=1.3.2 in /usr/local/lib/python3.9/dist-packages (from scikit-learn>=1.0.0->scikeras[tensorflow]) (1.10.1)
Requirement already satisfied: google-pasta>=0.1.1 in /usr/local/lib/python3.9/dist-packages (from tensorflow>=2.11.0->scikeras[tensorflow]) (0.2.0)
Requirement already satisfied: setuptools in /usr/local/lib/python3.9/dist-packages (from tensorflow>=2.11.0->scikeras[tensorflow]) (67.6.1)
Requirement already satisfied: libclang>=13.0.0 in /usr/local/lib/python3.9/dist-packages (from tensorflow>=2.11.0->scikeras[tensorflow]) (16.0.0)
Requirement already satisfied: h5py>=2.9.0 in /usr/local/lib/python3.9/dist-packages (from tensorflow>=2.11.0->scikeras[tensorflow]) (3.8.0)
Requirement already satisfied: gast<=0.4.0,>=0.2.1 in /usr/local/lib/python3.9/dist-packages (from tensorflow>=2.11.0->scikeras[tensorflow]) (0.4.0)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in /usr/local/lib/python3.9/dist-packages (from tensorflow>=2.11.0->scikeras[tensorflow]) (1.53.0)
Requirement already satisfied: keras<2.13,>=2.12.0 in /usr/local/lib/python3.9/dist-packages (from tensorflow>=2.11.0->scikeras[tensorflow]) (2.12.0)
Requirement already satisfied: flatbuffers>=2.0 in /usr/local/lib/python3.9/dist-packages (from tensorflow>=2.11.0->scikeras[tensorflow]) (23.3.3)
Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/python3.9/dist-packages (from tensorflow>=2.11.0->scikeras[tensorflow]) (2.2.0)
Requirement already satisfied: tensorboard<2.13,>=2.12 in /usr/local/lib/python3.9/dist-packages (from tensorflow>=2.11.0->scikeras[tensorflow]) (2.12.1)
Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.9/dist-packages (from tensorflow>=2.11.0->scikeras[tensorflow]) (1.16.0)
Requirement already satisfied: wrapt<1.15,>=1.11.0 in /usr/local/lib/python3.9/dist-packages (from tensorflow>=2.11.0->scikeras[tensorflow]) (1.14.1)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in /usr/local/lib/python3.9/dist-packages (from tensorflow>=2.11.0->scikeras[tensorflow]) (0.32.0)
Requirement already satisfied: astunparse>=1.6.0 in /usr/local/lib/python3.9/dist-packages (from tensorflow>=2.11.0->scikeras[tensorflow]) (1.6.3)
Requirement already satisfied: tensorflow-estimator<2.13,>=2.12.0 in /usr/local/lib/python3.9/dist-packages (from tensorflow>=2.11.0->scikeras[tensorflow]) (2.12.0)
Requirement already satisfied: absl-py>=1.0.0 in /usr/local/lib/python3.9/dist-packages (from tensorflow>=2.11.0->scikeras[tensorflow]) (1.4.0)
Requirement already satisfied: jax>=0.3.15 in /usr/local/lib/python3.9/dist-packages (from tensorflow>=2.11.0->scikeras[tensorflow]) (0.4.7)
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<5.0.0dev,>=3.20.3 in /usr/local/lib/python3.9/dist-packages (from tensorflow>=2.11.0->sci
Requirement already satisfied: opt-einsum>=2.3.2 in /usr/local/lib/python3.9/dist-packages (from tensorflow>=2.11.0->scikeras[tensorflow]) (3.3.0)
Requirement already satisfied: typing-extensions>=3.6.6 in /usr/local/lib/python3.9/dist-packages (from tensorflow>=2.11.0->scikeras[tensorflow]) (4.5.0)
Requirement already satisfied: wheel<1.0,>=0.23.0 in /usr/local/lib/python3.9/dist-packages (from astunparse>=1.6.0->tensorflow>=2.11.0->scikeras[tensorflow]) (0.40.0)
Requirement already satisfied: ml-dtypes>=0.0.3 in /usr/local/lib/python3.9/dist-packages (from jax>=0.3.15->tensorflow>=2.11.0->scikeras[tensorflow]) (0.0.4)
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.9/dist-packages (from tensorboard<2.13,>=2.12->tensorflow>=2.11.0->scikeras[tensorflow]) (3.4.3)
Requirement already satisfied: google-auth<3,>=1.6.3 in /usr/local/lib/python3.9/dist-packages (from tensorboard<2.13,>=2.12->tensorflow>=2.11.0->scikeras[tensorflow]) (2.17.2)
Requirement already satisfied: werkzeug>=1.0.1 in /usr/local/lib/python3.9/dist-packages (from tensorboard<2.13,>=2.12->tensorflow>=2.11.0->scikeras[tensorflow]) (2.2.3)
Requirement already satisfied: tensorboard-plugin-wit>=1.6.0 in /usr/local/lib/python3.9/dist-packages (from tensorboard<2.13,>=2.12->tensorflow>=2.11.0->scikeras[tensorflow]) (1.8.1)
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in /usr/local/lib/python3.9/dist-packages (from tensorboard<2.13,>=2.12->tensorflow>=2.11.0->scikeras[tensorflow])
Requirement already satisfied: google-auth-oauthlib<1.1,>=0.5 in /usr/local/lib/python3.9/dist-packages (from tensorboard<2.13,>=2.12->tensorflow>=2.11.0->scikeras[tensorflow]) (1.0.
Requirement already satisfied: requests<3,>=2.21.0 in /usr/local/lib/python3.9/dist-packages (from tensorboard<2.13,>=2.12->tensorflow>=2.11.0->scikeras[tensorflow]) (2.27.1)
Requirement already satisfied: cachetools<6.0,>=2.0.0 in /usr/local/lib/python3.9/dist-packages (from google-auth<3,>=1.6.3->tensorboard<2.13,>=2.12->tensorflow>=2.11.0->scikeras[tens
Requirement already satisfied: rsa<5,>=3.1.4 in /usr/local/lib/python3.9/dist-packages (from google-auth<3,>=1.6.3->tensorboard<2.13,>=2.12->tensorflow>=2.11.0->scikeras[tensorflow])
Requirement already satisfied: pyasn1-modules>=0.2.1 in /usr/local/lib/python3.9/dist-packages (from google-auth<3,>=1.6.3->tensorboard<2.13,>=2.12->tensorflow>=2.11.0->scikeras[tenso
Requirement already satisfied: requests-oauthlib>=0.7.0 in /usr/local/lib/python3.9/dist-packages (from google-auth-oauthlib<1.1,>=0.5->tensorboard<2.13,>=2.12->tensorflow>=2.11.0->sc
Requirement already satisfied: importlib-metadata>=4.4 in /usr/local/lib/python3.9/dist-packages (from markdown>=2.6.8->tensorboard<2.13,>=2.12->tensorflow>=2.11.0->scikeras[tensorflo
Requirement already satisfied: charset-normalizer~=2.0.0 in /usr/local/lib/python3.9/dist-packages (from requests<3,>=2.21.0->tensorboard<2.13,>=2.12->tensorflow>=2.11.0->scikeras[ter
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.9/dist-packages (from requests<3,>=2.21.0->tensorboard<2.13,>=2.12->tensorflow>=2.11.0->scikeras[tensorflow]) (3
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.9/dist-packages (from requests<3,>=2.21.0->tensorboard<2.13,>=2.12->tensorflow>=2.11.0->scikeras[tensorflow
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /usr/local/lib/python3.9/dist-packages (from requests<3,>=2.21.0->tensorboard<2.13,>=2.12->tensorflow>=2.11.0->scikeras[tensorf
Requirement already satisfied: MarkupSafe>=2.1.1 in /usr/local/lib/python3.9/dist-packages (from werkzeug>=1.0.1->tensorboard<2.13,>=2.12->tensorflow>=2.11.0->scikeras[tensorflow]) (2
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.9/dist-packages (from importlib-metadata>=4.4->markdown>=2.6.8->tensorboard<2.13,>=2.12->tensorflow>=2.11.0->scikera
Requirement already satisfied: pyasn1<0.5.0,>=0.4.6 in /usr/local/lib/python3.9/dist-packages (from pyasn1-modules>=0.2.1->google-auth<3,>=1.6.3->tensorboard<2.13,>=2.12->tensorflow>=
Requirement already satisfied: oauthlib>=3.0.0 in /usr/local/lib/python3.9/dist-packages (from requests-oauthlib>=0.7.0->google-auth-oauthlib<1.1,>=0.5->tensorboard<2.13,>=2.12->tenso
```

```
# importing modules

import numpy as np
import pandas as pd

from sklearn.model_selection import train_test_split, StratifiedKFold, cross_val_score
from sklearn.preprocessing import StandardScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
```

```
# load dataset
```

```
df = pd.read_csv("/content/drive/MyDrive/3rdYear/CPE019/hoa7.1/Bank_Personal_Loan_Modelling.csv")
dataset = df.values
```

```
df
```

	ID	Age	Experience	Income	ZIP Code	Family	CCAvg	Education	Mortgage	Personal Loan	Securities Account	CD Account	Online	C
0	1	25	1	49	91107	4	1.6	1	0	0	1	0	0	
1	2	45	19	34	90089	3	1.5	1	0	0	1	0	0	
2	3	39	15	11	94720	1	1.0	1	0	0	0	0	0	
3	4	35	9	100	94112	1	2.7	2	0	0	0	0	0	
4	5	35	8	45	91330	4	1.0	2	0	0	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
4995	4996	29	3	40	92697	1	1.9	3	0	0	0	0	1	
4996	4997	30	4	15	92037	4	0.4	1	85	0	0	0	1	
4997	4998	63	39	24	93023	2	0.3	3	0	0	0	0	0	
4998	4999	65	40	49	90034	3	0.5	2	0	0	0	0	1	
4999	5000	28	4	83	92612	3	0.8	1	0	0	0	0	1	

```
pip install scikeras
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: scikeras in /usr/local/lib/python3.9/dist-packages (0.10.0)
Requirement already satisfied: scikit-learn>=1.0.0 in /usr/local/lib/python3.9/dist-packages (from scikeras) (1.2.2)
Requirement already satisfied: packaging>=0.21 in /usr/local/lib/python3.9/dist-packages (from scikeras) (23.0)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.9/dist-packages (from scikit-learn>=1.0.0->scikeras) (1.2.0)
Requirement already satisfied: scipy>=1.3.2 in /usr/local/lib/python3.9/dist-packages (from scikit-learn>=1.0.0->scikeras) (1.10.1)
Requirement already satisfied: numpy>=1.17.3 in /usr/local/lib/python3.9/dist-packages (from scikit-learn>=1.0.0->scikeras) (1.22.4)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.9/dist-packages (from scikit-learn>=1.0.0->scikeras) (3.1.0)
```

```
# Separate the features and target variable
X = df.drop("Personal Loan", axis=1)
y = df["Personal Loan"]
```

```
y = df["Personal Loan"]

# Split the data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)



# Scale the data using StandardScaler
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)


# Create a base model with one hidden layer
def create_base_model():
    model = Sequential()
    model.add(Dense(20, input_dim=13, activation="relu"))
    model.add(Dense(1, activation="sigmoid"))
    model.compile(loss="binary_crossentropy", optimizer="adam", metrics=["accuracy"])
    return model


# Evaluate the model with k-fold cross-validation
kfold = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
estimator = KerasClassifier(build_fn=create_base_model, epochs=10, batch_size=32, verbose=0)
results = cross_val_score(estimator, X_train, y_train, cv=kfold)
print("Base model accuracy: %.2f%% (%.2f%%)" % (results.mean()*100, results.std()*100))
```

```
/usr/local/lib/python3.9/dist-packages/scikeras/wrappers.py:301: UserWarning: ``build_fn`` will be renamed to ``model`` in a future release, at which point use of ``build_fn`` will ra
  warnings.warn(
/usr/local/lib/python3.9/dist-packages/scikeras/wrappers.py:301: UserWarning: ``build_fn`` will be renamed to ``model`` in a future release, at which point use of ``build_fn`` will ra
  warnings.warn(
/usr/local/lib/python3.9/dist-packages/scikeras/wrappers.py:301: UserWarning: ``build_fn`` will be renamed to ``model`` in a future release, at which point use of ``build_fn`` will ra
  warnings.warn(
/usr/local/lib/python3.9/dist-packages/scikeras/wrappers.py:301: UserWarning: ``build_fn`` will be renamed to ``model`` in a future release, at which point use of ``build_fn`` will ra
  warnings.warn(
/usr/local/lib/python3.9/dist-packages/scikeras/wrappers.py:301: UserWarning: ``build_fn`` will be renamed to ``model`` in a future release, at which point use of ``build_fn`` will ra
  warnings.warn(
Base model accuracy: 96.00% (0.42%)
```

```
# Improve the accuracy of the model by adding additional hidden layers
def create_improved_model():
    model = Sequential()
    model.add(Dense(20, input_dim=13, activation="relu"))
    model.add(Dense(10, activation="relu"))
    model.add(Dense(5, activation="relu"))
    model.add(Dense(1, activation="sigmoid"))
    model.compile(loss="binary_crossentropy", optimizer="adam", metrics=["accuracy"])
    return model


# Evaluate the improved model with k-fold cross-validation
estimator = KerasClassifier(build_fn=create_improved_model, epochs=10, batch_size=32, verbose=0)
results = cross_val_score(estimator, X_train, y_train, cv=kfold)
print("Improved model accuracy: %.2f%% (%.2f%%)" % (results.mean()*100, results.std()*100))
```

```
/usr/local/lib/python3.9/dist-packages/scikeras/wrappers.py:301: UserWarning: ``build_fn`` will be renamed to ``model`` in a future release, at which point use of ``build_fn`` will ra
  warnings.warn(
/usr/local/lib/python3.9/dist-packages/scikeras/wrappers.py:301: UserWarning: ``build_fn`` will be renamed to ``model`` in a future release, at which point use of ``build_fn`` will ra
```

```
  warnings.warn(
/usr/local/lib/python3.9/dist-packages/scikeras/wrappers.py:301: UserWarning: ``build_fn`` will be renamed to ``model`` in a future release, at which point use of ``build_fn`` will ra
  warnings.warn(
/usr/local/lib/python3.9/dist-packages/scikeras/wrappers.py:301: UserWarning: ``build_fn`` will be renamed to ``model`` in a future release, at which point use of ``build_fn`` will ra
  warnings.warn(
/usr/local/lib/python3.9/dist-packages/scikeras/wrappers.py:301: UserWarning: ``build_fn`` will be renamed to ``model`` in a future release, at which point use of ``build_fn`` will ra
  warnings.warn(
Improved model accuracy: 96.70% (0.23%)
```

```
# Fit the best model on the entire training set and evaluate on the test set
best_model = create_improved_model()
best_model.fit(X_train, y_train, epochs=10, batch_size=32, verbose=0)
score = best_model.evaluate(X_test, y_test, verbose=0)
print("Test set accuracy: %.2f%%" % (score[1]*100))
```

```
Test set accuracy: 98.30%
```

Regression Problem

Title: California Housing Dataset

Link: https://www.kaggle.com/datasets/camnugent/california-housing-prices?resource=download

Explain your datasets and the problem being addressed.

The California Housing dataset includes data on the cost of homes in various Californian cities. The 20,640 instances and 10 attributes in this dataset include the median household income, median house value, latitude, and longitude. The objective of this dataset is to create a model that, depending on the other parameters, can forecast the median house value for a specific place. The outcome variable in this regression problem is a continuous numerical number. The dataset can be used to construct models that can forecast home prices in other regions as well as to study the relationship between different variables and Californian housing costs.

```
# importing modules

import numpy as np
import pandas as pd
```

```
path = "/content/drive/MyDrive/3rdYear/CPE019/hoa7.1/housing.csv"
df = pd.read_csv(path)
```

```
df
```

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	med:
0	-122.23	37.88	41.0	880.0	129.0	322.0	126.0	8.3252	
1	-122.22	37.86	21.0	7099.0	1106.0	2401.0	1138.0	8.3014	
2	-122.24	37.85	52.0	1467.0	190.0	496.0	177.0	7.2574	
3	-122.25	37.85	52.0	1274.0	235.0	558.0	219.0	5.6431	
4	-122.25	37.85	52.0	1627.0	280.0	565.0	259.0	3.8462	
...	...	...	...	...	...	...	...	...	
20635	-121.09	39.48	25.0	1665.0	374.0	845.0	330.0	1.5603	
20636	-121.21	39.49	18.0	697.0	150.0	356.0	114.0	2.5568	
20637	-121.22	39.43	17.0	2254.0	485.0	1007.0	433.0	1.7000	
20638	-121.32	39.43	18.0	1860.0	409.0	741.0	349.0	1.8672	
20639	-121.24	39.37	16.0	2785.0	616.0	1387.0	530.0	2.3886	

20640 rows × 10 columns

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import StandardScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping
from sklearn.impute import SimpleImputer
from sklearn.metrics import mean_squared_error, r2_score

# Load the dataset
df = pd.read_csv('/content/drive/MyDrive/3rdYear/CPE019/hoa7.1/housing.csv')

# One-hot encoding of categorical variables
df_encoded = pd.get_dummies(df, columns=['ocean_proximity'])

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(df_encoded.drop('median_house_value', axis=1), df['median_house_value'], test_size=0.2, random_state=42)

# Impute missing values in the test data
imputer = SimpleImputer(strategy='median')
X_test = imputer.fit_transform(X_test)

# Create a base linear regression model
model = LinearRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print(f'Base Model - Mean Squared Error: {mse:.2f}')
print(f'Base Model - R-Squared: {r2:.2f}')
```

```
Base Model - Mean Squared Error: 4909161624.07
Base Model - R-Squared: 0.63
/usr/local/lib/python3.9/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but LinearRegression was fitted with feature names
  warnings.warn(
```

```
# Standardize the training and testing sets
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Improve the model by standardizing the dataset
model_scaled = LinearRegression()
model_scaled.fit(X_train_scaled, y_train)
y_pred_scaled = model_scaled.predict(X_test_scaled)
mse_scaled = mean_squared_error(y_test, y_pred_scaled)
r2_scaled = r2_score(y_test, y_pred_scaled)
print(f'Standardized Model - Mean Squared Error: {mse_scaled:.2f}')
print(f'Standardized Model - R-Squared: {r2_scaled:.2f}')
```

```
Standardized Model - Mean Squared Error: 4909161624.07
Standardized Model - R-Squared: 0.63
/usr/local/lib/python3.9/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but StandardScaler was fitted with feature names
  warnings.warn(
```

```
# Define the neural network model
def create_model(input_dim, output_dim, hidden_layers, neurons):
    model = Sequential()
    model.add(Dense(neurons, input_dim=input_dim, activation='relu'))
    for i in range(hidden_layers):
        model.add(Dense(neurons, activation='relu'))
    model.add(Dense(output_dim))
    return model

# Evaluate small and large networks with varying layers and neurons
results = []
for hidden_layers in [1, 2, 3]:
    for neurons in [10, 50, 100]:
        model = create_model(input_dim=X_train_scaled.shape[1], output_dim=1, hidden_layers=hidden_layers, neurons=neurons)
        model.compile(loss='mse', optimizer=Adam(learning_rate=0.01))
        history = model.fit(X_train_scaled, y_train, validation_data=(X_test_scaled, y_test), epochs=100, batch_size=128, verbose=0)
        mse_small, r2_small = model_small.evaluate(X_test_scaled, y_test, verbose=0), r2_score(y_test, model_small.predict(X_test_scaled))
        results.append({'hidden_layers': hidden_layers, 'neurons': neurons, 'mse': mse, 'r2': r2})
        print(f'hidden_layers: {hidden_layers}, neurons: {neurons}, mse: {mse:.2f}, r2: {r2:.2f}')

# Print results
df_results = pd.DataFrame(results)
print(df_results)
```

```
129/129 [==============================] - 0s 1ms/step
hidden_layers: 1, neurons: 10, mse: 4909161624.07, r2: 0.63
129/129 [==============================] - 0s 850us/step
hidden_layers: 1, neurons: 50, mse: 4909161624.07, r2: 0.63
129/129 [==============================] - 0s 849us/step
hidden_layers: 1, neurons: 100, mse: 4909161624.07, r2: 0.63
129/129 [==============================] - 0s 831us/step
```

```
hidden_layers: 2, neurons: 10, mse: 4909161624.07, r2: 0.63
129/129 [==============================] - 0s 922us/step
hidden_layers: 2, neurons: 50, mse: 4909161624.07, r2: 0.63
129/129 [==============================] - 0s 893us/step
hidden_layers: 2, neurons: 100, mse: 4909161624.07, r2: 0.63
129/129 [==============================] - 0s 3ms/step
hidden_layers: 3, neurons: 10, mse: 4909161624.07, r2: 0.63
129/129 [==============================] - 0s 1ms/step
hidden_layers: 3, neurons: 50, mse: 4909161624.07, r2: 0.63
129/129 [==============================] - 0s 1ms/step
hidden_layers: 3, neurons: 100, mse: 4909161624.07, r2: 0.63
   hidden_layers  neurons           mse        r2
0              1       10  4.909162e+09  0.625372
1              1       50  4.909162e+09  0.625372
2              1      100  4.909162e+09  0.625372
3              2       10  4.909162e+09  0.625372
4              2       50  4.909162e+09  0.625372
5              2      100  4.909162e+09  0.625372
6              3       10  4.909162e+09  0.625372
7              3       50  4.909162e+09  0.625372
8              3      100  4.909162e+09  0.625372
```

```
results = []
for hidden_layers in [1, 2, 3]:
    for neurons in [10, 50, 100]:
        model_large = create_model(input_dim=X_train_scaled.shape[1], output_dim=1, hidden_layers=hidden_layers, neurons=neurons)
        model_large.compile(loss='mse', optimizer=Adam(learning_rate=0.01))
        history_large = model_large.fit(X_train_scaled, y_train, validation_data=(X_test_scaled, y_test), epochs=100, batch_size=128, verbose=0)
        mse_large, r2_large = model_large.evaluate(X_test_scaled, y_test, verbose=0), r2_score(y_test, model_large.predict(X_test_scaled))
        results.append({'hidden_layers': hidden_layers, 'neurons': neurons, 'mse': mse, 'r2': r2})
        print(f'hidden_layers: {hidden_layers}, neurons: {neurons}, mse: {mse:.2f}, r2: {r2:.2f}')

# Print results
df_results = pd.DataFrame(results)
print(df_results)
```

```
129/129 [==============================] - 0s 968us/step
hidden_layers: 1, neurons: 10, mse: 4909161624.07, r2: 0.63
129/129 [==============================] - 0s 830us/step
hidden_layers: 1, neurons: 50, mse: 4909161624.07, r2: 0.63
129/129 [==============================] - 0s 886us/step
hidden_layers: 1, neurons: 100, mse: 4909161624.07, r2: 0.63
129/129 [==============================] - 0s 1ms/step
hidden_layers: 2, neurons: 10, mse: 4909161624.07, r2: 0.63
129/129 [==============================] - 0s 843us/step
hidden_layers: 2, neurons: 50, mse: 4909161624.07, r2: 0.63
129/129 [==============================] - 0s 1ms/step
hidden_layers: 2, neurons: 100, mse: 4909161624.07, r2: 0.63
129/129 [==============================] - 0s 895us/step
hidden_layers: 3, neurons: 10, mse: 4909161624.07, r2: 0.63
129/129 [==============================] - 0s 972us/step
hidden_layers: 3, neurons: 50, mse: 4909161624.07, r2: 0.63
129/129 [==============================] - 0s 937us/step
hidden_layers: 3, neurons: 100, mse: 4909161624.07, r2: 0.63
   hidden_layers  neurons           mse        r2
0              1       10  4.909162e+09  0.625372
1              1       50  4.909162e+09  0.625372
2              1      100  4.909162e+09  0.625372
3              2       10  4.909162e+09  0.625372
```

```
4              2       50  4.909162e+09  0.625372
5              2      100  4.909162e+09  0.625372
6              3       10  4.909162e+09  0.625372
7              3       50  4.909162e+09  0.625372
8              3      100  4.909162e+09  0.625372
```

Conclusion

In conclusion, the student successfully addressed errors such as "ValueError: Input X contains NaN" and "ValueError: could not convert string to float: 'NEAR OCEAN'." They created a Python program for both classification and regression problems. Additionally, the student demonstrated the ability to determine whether a dataset is suitable for a classification or regression problem.

Submit the link to your Google Colab (make sure that it is accessible to me)

Link: https://colab.research.google.com/drive/14lXmBBpnBIA3FR-j_tjT6Or6tizKNhjb?usp=sharing