In [ ]:

```
# Reading of the datasets and understanding the size,the no. of null values and various other elements

import pandas as pd

gt =  pd.read_csv("datasets/Genotypic_Data.csv")
pt=pd.read_csv("datasets\Phenotypic_Data.csv")

pt.size
gt.size
pt.head()
gt.head()

gt.info()
pt.info()
gt.isnull().sum()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14790 entries, 0 to 14789
Columns: 291 entries, SNPs to WH1142
dtypes: float64(5), int64(1), object(285)
memory usage: 32.8+ MB
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 280 entries, 0 to 279
Data columns (total 32 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   Genotype             280 non-null    object
 1   DH_Dharwad           280 non-null    float64
 2   DH_IARI-DELHI        280 non-null    float64
 3   DH_IARI-Jharkhand    280 non-null    float64
 4   DH_KARNAL            280 non-null    float64
 5   DH_Pooled            280 non-null    float64
 6   GFD_Dharwad          280 non-null    float64
 7   GFD_IARI-Delhi       280 non-null    float64
 8   GFD_IARI-Jharkhand   280 non-null    float64
 9   GFD_Karnal           280 non-null    float64
 10  GFD_Pooled           280 non-null    float64
 11  GNPS_Dharwad         280 non-null    float64
 12  GNPS_IARI-Jharkhand  280 non-null    float64
 13  GNPS_Pooled          280 non-null    float64
 14  GWPS_Dharwad         280 non-null    float64
 15  GWPS_IARI-Delhi      280 non-null    float64
 16  GWPS_IARI-Jharkhand  280 non-null    float64
 17  GWPS_Karnal          280 non-null    float64
 18  GWPS_Ludhiana        280 non-null    float64
 19  GWPS_Pooled          280 non-null    float64
 20  PH_Dharwad           280 non-null    float64
 21  PH_IARI-Delhi        280 non-null    float64
 22  PH_IARI-Jharkhand    280 non-null    float64
 23  PH_Karnal            280 non-null    float64
 24  PH_Ludhiana          280 non-null    float64
 25  PH_Pooled            280 non-null    float64
 26  GY_Dharwad           280 non-null    float64
 27  GY_IARI-Delhi        280 non-null    float64
 28  GY_IARI-JKD          280 non-null    float64
 29  GY_Karnal            280 non-null    float64
 30  GY_Ludhiana          280 non-null    float64
```

```
 31  GY_Pooled            280 non-null    float64
dtypes: float64(31), object(1)
memory usage: 70.1+ KB
```

Out[ ]:
```
SNPs          0
alleles       0
Chrom         0
Pos           0
strand        0
             ...
DBW173      379
DBW187      256
MACS6222    265
WH1124      206
WH1142      283
Length: 291, dtype: int64
```

In [ ]:
```
# Removing the useless or no related cols from the dataset
col=pt['Genotype']
cols_to_drop=['Chrom','Pos','strand','assembly','center','protLSID','assayLSID','panel','QCcode']
gt=gt.drop(cols_to_drop,axis=1)
gt.to_csv("updated_genotype",index=False)
```

In [ ]:
```
gt.head()
```

Out[ ]:

	SNPs	alleles	AAI-W29	AKAW5080	AKAW5099	BRW3877	CG1029	CG1034	CG1035	CG1036	...	HD3372	HI1655	AKAW5088	MA
0	AX-94381285	C/T	CC	CC	CC	CC	NaN	TC	CC	CC	...	CC	NaN	CC	
1	AX-94383718	A/G	AA	AA	AA	AA	AA	AA	AA	AA	...	AA	AA	AA	
2	AX-94384181	A/G	NaN	NaN	AA	AG	GG	AA	NaN	GG	...	AA	GG	AA	
3	AX-94384966	T/C	TT	CC	TT	TC	TT	TT	TT	CC	...	TT	TT	CC	
4	AX-94386458	C/A	AA	AC	CC	CC	AC	AA	AA	NaN	...	AA	CC	CC	

5 rows × 282 columns

In [ ]:
```
#Checking for same genotypes in both datasets

import numpy as np
gt_cid=gt.columns[2:].to_list()
pt_cid=pt['Genotype'].to_list()
sorted(gt_cid)==sorted(pt_cid)
```

Out[ ]:
```
True
```

In [ ]:
```
df= pd.read_csv("updated_genotype")
df.isnull().sum()
```

Out[ ]:
```
SNPs           0
alleles        0
AAI-W29     1155
AKAW5080     624
AKAW5099     616
             ...
DBW173       379
DBW187       256
MACS6222     265
WH1124       206
WH1142       283
Length: 282, dtype: int64
```

In [ ]:
```python
# Replacing the null values for given dataset with primary allele

import pandas as pd

def replace_nan_with_first_letter(value, alleles):
    if pd.isna(value):
        letter_before_slash = alleles.split('/')[0]
        return letter_before_slash * 2
    else:
        return value

for column in df.columns[1:]:
    df[column] = df.apply(lambda row: replace_nan_with_first_letter(row[column], row['alleles']), axis=1)

df=df.drop(['alleles'],axis=1)
df.to_csv("final_genotype",index=False)
```

In [ ]:
```python
# Removing the other cols from phenotype dataset as we consider only pooled values

dropme=[]
for i in pt.columns:
    if i=='DH_Pooled' or i=='GFD_Pooled' or i=='GNPS_Pooled' or i=='GWPS_Pooled' or i=='PH_Pooled' or i=='GY_Pooled':
        continue
    else:
        dropme.append(i)
dropme=dropme[1:]
```

```
pt=pt.drop(dropme,axis=1)
pt.to_csv("final_phenotype",index=False)
```

In [ ]:
```
# Label Encoding and replacing the encoded values

geno_map={'AA':1,'AT':2,'AG':3,'AC':4,'TT':5,'TG':6,'TC':7,'GG':8,'CG':9,'CC':10}
sample_geno=pd.read_csv("final_genotype")
df2=pd.DataFrame(sample_geno)

for i in df2[1:]:
    df2[i]=df2[i].map(geno_map)

df2['SNPs']=df['SNPs']
df2.head()
df2.to_csv("fgenotype",index=False)
```

In [ ]:
```
# Merging both the dataset based on genotype

import pandas as pd

fg=pd.read_csv("fgenotype")
fp=pd.read_csv("final_phenotype")



fg = fg.set_index('SNPs').T.reset_index()

fg.columns.name = None
fg = fg.rename(columns={'index': 'Genotype'})

merged_dataset = pd.merge(fg, fp, on='Genotype')

merged_dataset.to_csv("mergeds",index=False)
```

In [ ]:
```
# Finding the Coorelation among the snp data and traits(pooled values) and Finding the number of common snps which highly infl
p=[]
f=[]
import pandas as pd
poled=[ 'DH_Pooled','GFD_Pooled' ,'GNPS_Pooled','GWPS_Pooled','PH_Pooled','GY_Pooled']
```

```
md=pd.read_csv("mergeds")

snp_data = md.iloc[:, 1:14790]

for i in poled:
    gfd_pooled = md[i]
    corr_gfd= snp_data.corrwith(gfd_pooled)
    snp_corr_df = pd.DataFrame({'SNP': snp_data.columns, 'Correlation': corr_gfd})
    snp_corr_df = snp_corr_df.sort_values(by='Correlation', ascending=False)

    k=[]

    for i ,j in  enumerate(snp_corr_df['Correlation']):
        if(j>0):
            k.append(snp_corr_df['SNP'].iloc[i])
    k = k[0:len(k)*4//5]
    p.append(k)


for i in p:
    print(len(i))
f = p
list1 = f[0]
list2 = f[1]
list3 = f[2]
list4 = f[3]
list5 = f[4]
list6 = f[5]




set1 = set(list1)
set2 = set(list2)
set3 = set(list3)
set4 = set(list4)
set5 = set(list5)
set6 = set(list6)
# Find the common elements
common_elements = set1.intersection(set2,set3,set4,set5,set6)
#common_elements = set(common_elements).intersection(set3)
```

```
# Convert the result back to a list (if needed)
common_elements_list = list(common_elements)

print(len(common_elements_list))
```

```
5878
6084
5863
5925
6052
5869
24
```

In [ ]:
```
# Removing the non related snps from the dataset
dp=[]
for i in sorted(md.columns[1:]):
    if i not in common_elements_list:
        dp.append(i)

poled=[ 'DH_Pooled','GFD_Pooled' ,'GNPS_Pooled','GWPS_Pooled','PH_Pooled','GY_Pooled']
for i in poled:
    dp.remove(i)
```

In [ ]:
```
# Reading the Final dataset

md1=pd.read_csv("mergeds")
md1=md1.drop(columns=dp)
md1.to_csv("pheno_geno",index=False)
```

In [ ]:
```
len(md1['Genotype'])
```

Out[ ]:
```
280
```

In [ ]:
```
# # Building the DL model ->(FNN model) and Predicting the values

# import numpy as np
# import tensorflow as tf
# from tensorflow import keras
```

```
# from sklearn.model_selection import train_test_split
# from sklearn.preprocessing import MinMaxScaler
# from sklearn.metrics import mean_squared_error

# # Load your dataset
# # Assuming df contains your data
# df = md1

# # X should be the genetic sequences
# # y should be the trait values (DH_Pooled, GFD_Pooled, GNPS_Pooled, GWPS_Pooled, PH_Pooled, GY_Pooled)
# X = df.iloc[:, 1:25]  # Assuming the DNA sequences start from the second column

# # Extract the trait values
# y = df[['DH_Pooled', 'GFD_Pooled', 'GNPS_Pooled', 'GWPS_Pooled', 'PH_Pooled', 'GY_Pooled']]

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

# # Standardize input features and trait values separately
# scaler_X = MinMaxScaler()
# scaler_y = MinMaxScaler()

# X_train_scaled = scaler_X.fit_transform(X_train)
# X_test_scaled = scaler_X.transform(X_test)
# y_train_scaled = scaler_y.fit_transform(y_train)
# y_test_scaled = scaler_y.transform(y_test)

# # Build the FNN model
# model = keras.Sequential([
#     keras.layers.Input(shape=(X_train.shape[1],)),
#     keras.layers.Dense(128, activation='elu'),
#     keras.layers.Dense(64, activation='elu'),
#     keras.layers.Dense(32, activation='elu'),
#     keras.layers.Dense(6, activation='sigmoid')  # Use sigmoid activation for output layer for values between 0 and 1
# ])

# # Compile the model
# model.compile(optimizer='adam', loss='mean_squared_error')

# # Train the model
# model.fit(X_train_scaled, y_train_scaled, epochs=50, batch_size=10, validation_split=0.2)
```

```
# # Evaluate the model on the test set
# y_pred_scaled = model.predict(X_test_scaled)
# mae = mean_squared_error(y_test_scaled, y_pred_scaled)

# # Generate and predict a new sequence
# new_sequence_1 = np.array([1,2,3,4,5,6,7,8,9,10,1,2,3,4,5,6,7,8,9,1,2,3,4,4]).reshape(1, -1)  # Reshape to match input shape
# scaled_new_sequence = scaler_X.transform(new_sequence_1)
# predictions_scaled = model.predict(scaled_new_sequence)

# # Inverse transform the scaled predictions to get the original scale
# predictions = scaler_y.inverse_transform(predictions_scaled)

# print("Root Mean Squared Error (RMSE):", np.sqrt(mae))
# print("Predictions:", predictions)
```

```
WARNING:tensorflow:From c:\Users\chris\AppData\Local\Programs\Python\Python311\Lib\site-packages\keras\src\losses.py:2976: The
name tf.losses.sparse_softmax_cross_entropy is deprecated. Please use tf.compat.v1.losses.sparse_softmax_cross_entropy instead.

WARNING:tensorflow:From c:\Users\chris\AppData\Local\Programs\Python\Python311\Lib\site-packages\keras\src\backend.py:1398: The
name tf.executing_eagerly_outside_functions is deprecated. Please use tf.compat.v1.executing_eagerly_outside_functions instead.

WARNING:tensorflow:From c:\Users\chris\AppData\Local\Programs\Python\Python311\Lib\site-packages\keras\src\optimizers\__init__.
py:309: The name tf.train.Optimizer is deprecated. Please use tf.compat.v1.train.Optimizer instead.

Epoch 1/50
WARNING:tensorflow:From c:\Users\chris\AppData\Local\Programs\Python\Python311\Lib\site-packages\keras\src\utils\tf_utils.py:49
2: The name tf.ragged.RaggedTensorValue is deprecated. Please use tf.compat.v1.ragged.RaggedTensorValue instead.

18/18 [==============================] - 1s 15ms/step - loss: 0.0304 - val_loss: 0.0222
Epoch 2/50
18/18 [==============================] - 0s 5ms/step - loss: 0.0239 - val_loss: 0.0218
Epoch 3/50
18/18 [==============================] - 0s 5ms/step - loss: 0.0226 - val_loss: 0.0206
Epoch 4/50
18/18 [==============================] - 0s 5ms/step - loss: 0.0221 - val_loss: 0.0209
Epoch 5/50
18/18 [==============================] - 0s 6ms/step - loss: 0.0221 - val_loss: 0.0221
Epoch 6/50
18/18 [==============================] - 0s 5ms/step - loss: 0.0222 - val_loss: 0.0209
Epoch 7/50
18/18 [==============================] - 0s 5ms/step - loss: 0.0219 - val_loss: 0.0209
Epoch 8/50
18/18 [==============================] - 0s 5ms/step - loss: 0.0213 - val_loss: 0.0212
Epoch 9/50
18/18 [==============================] - 0s 5ms/step - loss: 0.0213 - val_loss: 0.0217
Epoch 10/50
18/18 [==============================] - 0s 5ms/step - loss: 0.0214 - val_loss: 0.0208
Epoch 11/50
18/18 [==============================] - 0s 5ms/step - loss: 0.0210 - val_loss: 0.0216
Epoch 12/50
18/18 [==============================] - 0s 7ms/step - loss: 0.0213 - val_loss: 0.0225
Epoch 13/50
18/18 [==============================] - 0s 7ms/step - loss: 0.0213 - val_loss: 0.0224
Epoch 14/50
18/18 [==============================] - 0s 6ms/step - loss: 0.0211 - val_loss: 0.0214
Epoch 15/50
```

```
18/18 [==============================] - 0s 5ms/step - loss: 0.0211 - val_loss: 0.0213
Epoch 16/50
18/18 [==============================] - 0s 6ms/step - loss: 0.0212 - val_loss: 0.0214
Epoch 17/50
18/18 [==============================] - 0s 7ms/step - loss: 0.0207 - val_loss: 0.0227
Epoch 18/50
18/18 [==============================] - 0s 6ms/step - loss: 0.0203 - val_loss: 0.0215
Epoch 19/50
18/18 [==============================] - 0s 5ms/step - loss: 0.0202 - val_loss: 0.0216
Epoch 20/50
18/18 [==============================] - 0s 6ms/step - loss: 0.0200 - val_loss: 0.0220
Epoch 21/50
18/18 [==============================] - 0s 7ms/step - loss: 0.0200 - val_loss: 0.0232
Epoch 22/50
18/18 [==============================] - 0s 6ms/step - loss: 0.0216 - val_loss: 0.0217
Epoch 23/50
18/18 [==============================] - 0s 5ms/step - loss: 0.0202 - val_loss: 0.0213
Epoch 24/50
18/18 [==============================] - 0s 5ms/step - loss: 0.0194 - val_loss: 0.0224
Epoch 25/50
18/18 [==============================] - 0s 5ms/step - loss: 0.0196 - val_loss: 0.0219
Epoch 26/50
18/18 [==============================] - 0s 4ms/step - loss: 0.0193 - val_loss: 0.0222
Epoch 27/50
18/18 [==============================] - 0s 5ms/step - loss: 0.0193 - val_loss: 0.0224
Epoch 28/50
18/18 [==============================] - 0s 5ms/step - loss: 0.0189 - val_loss: 0.0227
Epoch 29/50
18/18 [==============================] - 0s 5ms/step - loss: 0.0192 - val_loss: 0.0223
Epoch 30/50
18/18 [==============================] - 0s 5ms/step - loss: 0.0189 - val_loss: 0.0230
Epoch 31/50
18/18 [==============================] - 0s 4ms/step - loss: 0.0186 - val_loss: 0.0223
Epoch 32/50
18/18 [==============================] - 0s 5ms/step - loss: 0.0189 - val_loss: 0.0226
Epoch 33/50
18/18 [==============================] - 0s 5ms/step - loss: 0.0187 - val_loss: 0.0236
Epoch 34/50
18/18 [==============================] - 0s 5ms/step - loss: 0.0186 - val_loss: 0.0233
Epoch 35/50
18/18 [==============================] - 0s 4ms/step - loss: 0.0187 - val_loss: 0.0232
```

```
Epoch 36/50
18/18 [==============================] - 0s 5ms/step - loss: 0.0185 - val_loss: 0.0230
Epoch 37/50
18/18 [==============================] - 0s 5ms/step - loss: 0.0176 - val_loss: 0.0234
Epoch 38/50
18/18 [==============================] - 0s 5ms/step - loss: 0.0178 - val_loss: 0.0237
Epoch 39/50
18/18 [==============================] - 0s 5ms/step - loss: 0.0178 - val_loss: 0.0236
Epoch 40/50
18/18 [==============================] - 0s 7ms/step - loss: 0.0169 - val_loss: 0.0240
Epoch 41/50
18/18 [==============================] - 0s 6ms/step - loss: 0.0168 - val_loss: 0.0241
Epoch 42/50
18/18 [==============================] - 0s 5ms/step - loss: 0.0170 - val_loss: 0.0242
Epoch 43/50
18/18 [==============================] - 0s 5ms/step - loss: 0.0169 - val_loss: 0.0241
Epoch 44/50
18/18 [==============================] - 0s 5ms/step - loss: 0.0171 - val_loss: 0.0253
Epoch 45/50
18/18 [==============================] - 0s 5ms/step - loss: 0.0164 - val_loss: 0.0254
Epoch 46/50
18/18 [==============================] - 0s 4ms/step - loss: 0.0163 - val_loss: 0.0255
Epoch 47/50
18/18 [==============================] - 0s 5ms/step - loss: 0.0160 - val_loss: 0.0250
Epoch 48/50
18/18 [==============================] - 0s 5ms/step - loss: 0.0162 - val_loss: 0.0257
Epoch 49/50
18/18 [==============================] - 0s 5ms/step - loss: 0.0165 - val_loss: 0.0271
Epoch 50/50
18/18 [==============================] - 0s 5ms/step - loss: 0.0155 - val_loss: 0.0259
2/2 [==============================] - 0s 2ms/step
1/1 [==============================] - 0s 27ms/step
Root Mean Squared Error (RMSE): 0.1543574068244376
Predictions: [[ 97.206604   43.037956   50.916977    2.0546627 109.24772   311.618    ]]
c:\Users\chris\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\base.py:464: UserWarning: X does not have vali
d feature names, but MinMaxScaler was fitted with feature names
  warnings.warn(
```

In [ ]:
```
import numpy as np
import pandas as pd
```

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error

# Load your dataset
# Assuming df contains your data
df = md1

# X should be the genetic sequences
# y should be the trait values (DH_Pooled, GFD_Pooled, GNPS_Pooled, GWPS_Pooled, PH_Pooled, GY_Pooled)
X = df.iloc[:, 1:25]  # Assuming the DNA sequences start from the second column

# Extract the trait values
y = df[['DH_Pooled', 'GFD_Pooled', 'GNPS_Pooled', 'GWPS_Pooled', 'PH_Pooled', 'GY_Pooled']]

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

# Standardize input features and trait values separately (using the same scalers)
scaler_X = MinMaxScaler()
scaler_y = MinMaxScaler()

X_train_scaled = scaler_X.fit_transform(X_train)
X_test_scaled = scaler_X.transform(X_test)
y_train_scaled = scaler_y.fit_transform(y_train)
y_test_scaled = scaler_y.transform(y_test)

# Build and train a Random Forest Regressor model
random_forest_model = RandomForestRegressor(n_estimators=100, random_state=42)  # You can adjust hyperparameters
random_forest_model.fit(X_train_scaled, y_train_scaled)

# Predict on the test set
y_pred_scaled = random_forest_model.predict(X_test_scaled)

# Calculate Mean Squared Error
mae = mean_squared_error(y_test_scaled, y_pred_scaled)

# Generate and predict a new sequence
new_sequence_1 = np.array([1,2,3,4,5,6,7,8,9,10,1,2,3,4,5,6,7,8,9,1,2,3,4,4]).reshape(1, -1)  # Reshape to match input shape
scaled_new_sequence = scaler_X.transform(new_sequence_1)
```

```
predictions_scaled = random_forest_model.predict(scaled_new_sequence)

# Inverse transform the scaled predictions to get the original scale
predictions = scaler_y.inverse_transform(predictions_scaled)

print("Mean Squared Error (MSE):", mae)
print("Predictions:", predictions)
```

```
Mean Squared Error (MSE): 0.023836786555106307
Predictions: [[ 84.16381667  43.01147667  48.46255667   2.25056667  96.10787
  429.26648333]]
```

```
c:\Users\chris\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\base.py:464: UserWarning: X does not have vali
d feature names, but MinMaxScaler was fitted with feature names
  warnings.warn(
```

In [ ]:
```
import joblib

# Save the Random Forest model
joblib.dump(random_forest_model, 'random_forest_model.pkl')

# Save the scalers
joblib.dump(scaler_X, 'scaler_X.pkl')
joblib.dump(scaler_y, 'scaler_y.pkl')
```

Out[ ]:
```
['scaler_y.pkl']
```

In [ ]:
```
# import tensorflow as tf

# # Load your Keras model
# model = tf.keras.models.load_model('Final_Model')

# # Convert the Keras model to TFLite
# converter = tf.lite.TFLiteConverter.from_keras_model(model)
# tflite_model = converter.convert()

# # Save the TFLite model to a file
# with open('TFLYT\FNN_quant.tflite', 'wb') as f:
#     f.write(tflite_model)
```

In [ ]:

```
# import pandas as pd
# gg=pd.read_csv("datasets\Genotypic_Data.csv")
# kk=pd.read_csv("pheno_geno")

# dp=[]
# for i in kk.columns:
#     dp.append(i)

# dp=dp[1:]
# dp=dp[:24]

# snps_series = pd.Series(df['SNPs'])
# df=gg
# pos_values = df[df['SNPs'].isin(dp)]['Pos']



# print(pos_values.tolist())
```