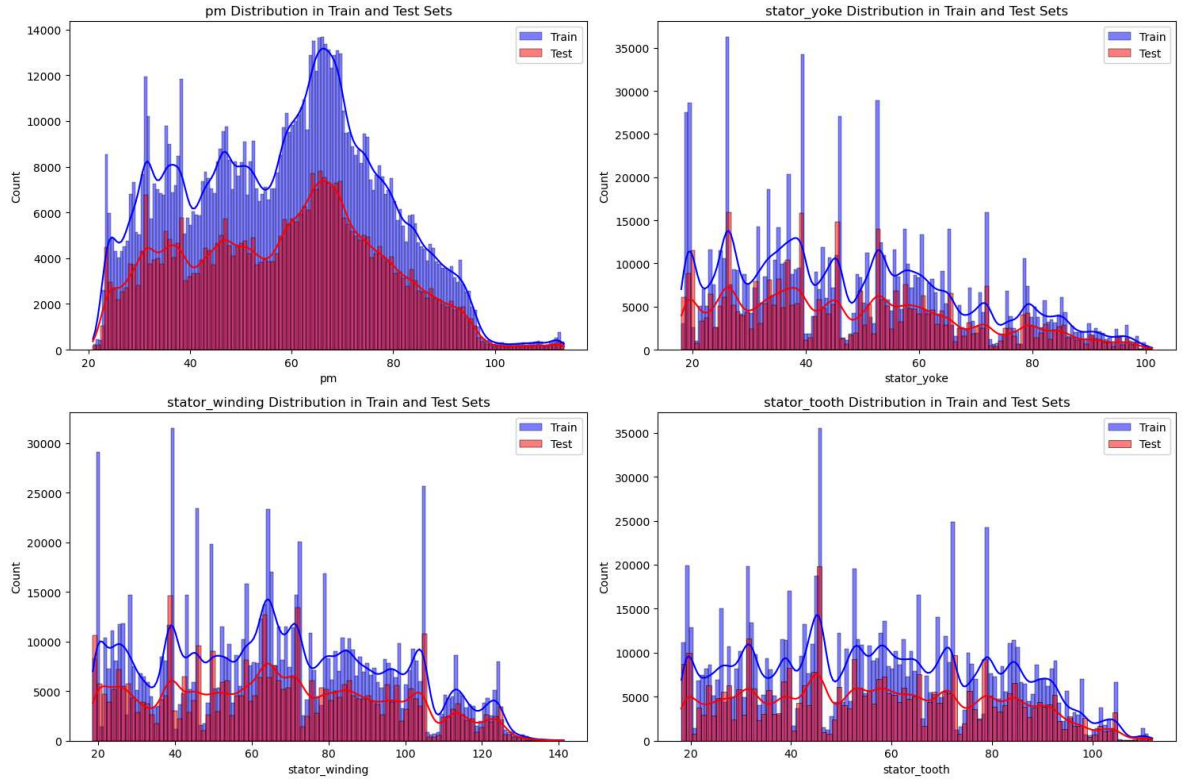
In [10]:
```
import tensorflow as tf
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler

# Suppress unnecessary warnings
import warnings
warnings.filterwarnings("ignore")

# Load dataset (replace with actual file path)
data_path = "C:\\Users\\User\\Downloads\\measures_v2.csv\\measures_v2.csv"
data = pd.read_csv(data_path)

# Display the first few rows of the dataset
print(data.head())

# Visualize the correlation matrix
corr = data.corr()
colormap = sns.diverging_palette(220, 10, as_cmap=True)
plt.figure(figsize=(10, 8))
sns.heatmap(corr, cbar=True, square=True, annot=True, fmt='.2f',
            cmap=colormap, linewidths=0.1, linecolor='white')
plt.title('Correlation of Features', y=1.05, size=15)
plt.show()

# Split dataset into train and test
train, test = train_test_split(data, test_size=0.3, random_state=42)
print("Training Data:", train.shape)
print("Testing Data:", test.shape)

# Separate input features and output labels
input_features = ['ambient', 'coolant', 'u_d', 'u_q', 'motor_speed', 'torque'
output_labels = ['pm', 'stator_yoke', 'stator_winding', 'stator_tooth']

train_x = train[input_features]
test_x = test[input_features]
train_y = train[output_labels]
test_y = test[output_labels]

# Normalize the input features
scaler = MinMaxScaler()
train_x = scaler.fit_transform(train_x)
test_x = scaler.transform(test_x)

# Define the model
model = tf.keras.Sequential([
    tf.keras.layers.Input(shape=(train_x.shape[1],)),
    tf.keras.layers.Dense(units=train_y.shape[1], activation='sigmoid')
])

# Compile the model
model.compile(optimizer=tf.keras.optimizers.SGD(learning_rate=0.1),
              loss='binary_crossentropy',
              metrics=['accuracy'])
```

```
# Train the model
history = model.fit(train_x, train_y, epochs=100, batch_size=32, validation_s

# Evaluate the model
train_loss, train_acc = model.evaluate(train_x, train_y, verbose=2)
test_loss, test_acc = model.evaluate(test_x, test_y, verbose=2)
print("Training Accuracy:", train_acc)
print("Testing Accuracy:", test_acc)

# Plot training & validation accuracy values
plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend(['Train', 'Validation'], loc='upper left')

# Plot training & validation loss values
plt.subplot(1, 2, 2)
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend(['Train', 'Validation'], loc='upper left')
plt.show()
```

```
23290/23290 - 15s - 633us/step - accuracy: 0.6453 - loss: -4.5973e+08 - v
al_accuracy: 0.6444 - val_loss: -4.6284e+08
Epoch 74/100
23290/23290 - 16s - 672us/step - accuracy: 0.6453 - loss: -4.6608e+08 - v
al_accuracy: 0.6444 - val_loss: -4.6919e+08
Epoch 75/100
23290/23290 - 16s - 684us/step - accuracy: 0.6453 - loss: -4.7242e+08 - v
al_accuracy: 0.6444 - val_loss: -4.7553e+08
Epoch 76/100
23290/23290 - 17s - 713us/step - accuracy: 0.6453 - loss: -4.7876e+08 - v
al_accuracy: 0.6444 - val_loss: -4.8187e+08
Epoch 77/100
23290/23290 - 16s - 679us/step - accuracy: 0.6453 - loss: -4.8511e+08 - v
al_accuracy: 0.6444 - val_loss: -4.8822e+08
Epoch 78/100
23290/23290 - 15s - 661us/step - accuracy: 0.6453 - loss: -4.9145e+08 - v
al_accuracy: 0.6444 - val_loss: -4.9456e+08
Epoch 79/100
23290/23290 - 16s - 690us/step - accuracy: 0.6453 - loss: -4.9780e+08 - v
al_accuracy: 0.6444 - val_loss: -5.0091e+08
```

In [15]:

```
# Visualize the distribution of output labels in training and testing sets
plt.figure(figsize=(15, 10))
for i, label in enumerate(output_labels):
    plt.subplot(2, 2, i+1)
    sns.histplot(train_y[label], kde=True, color='blue', label='Train')
    sns.histplot(test_y[label], kde=True, color='red', label='Test', alpha=0.
    plt.title(f'{label} Distribution in Train and Test Sets')
    plt.legend()
plt.tight_layout()
plt.show()
```



In [ ]: