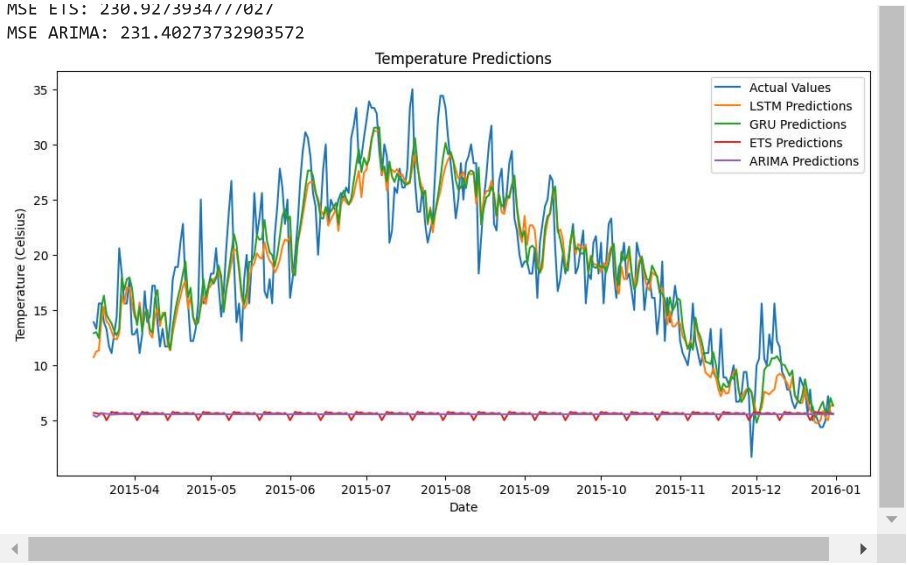
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, GRU, Dense
from statsmodels.tsa.holtwinters import ExponentialSmoothing
from statsmodels.tsa.arima.model import ARIMA
from sklearn.metrics import mean_squared_error

# Load data
file_path = '/content/seattle-weather.csv'
data = pd.read_csv(file_path)

# Convert 'date' to datetime
data['date'] = pd.to_datetime(data['date'])

# Extract time-based features
data['month'] = data['date'].dt.month
data['day'] = data['date'].dt.day
data['hour'] = data['date'].dt.hour
data['day_of_week'] = data['date'].dt.dayofweek

# Fill missing values
data = data.fillna(method='ffill')

# Encode categorical 'weather' feature
data = pd.get_dummies(data, columns=['weather'])

# Select relevant columns for scaling
features = ['temp_max', 'temp_min', 'precipitation', 'wind', 'month', 'day', 'hour', 'day_of_week'] + [col for col in data.columns if col.sta
scaler = MinMaxScaler()
scaled_data = scaler.fit_transform(data[features])

# Create sequences
def create_sequences(data, look_back=10):
    X, y = [], []
    for i in range(len(data) - look_back):
        X.append(data[i:(i + look_back)])
        y.append(data[i + look_back, 0])  # Predict temp_max
    return np.array(X), np.array(y)

look_back = 10
X, y = create_sequences(scaled_data, look_back)

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, shuffle=False)

# Build and train LSTM model
lstm_model = Sequential()
lstm_model.add(LSTM(50, input_shape=(look_back, X.shape[2])))
lstm_model.add(Dense(1))
lstm_model.compile(optimizer='adam', loss='mean_squared_error')
lstm_model.fit(X_train, y_train, epochs=20, batch_size=32, validation_split=0.2)

# Build and train GRU model
gru_model = Sequential()
gru_model.add(GRU(50, input_shape=(look_back, X.shape[2])))
gru_model.add(Dense(1))
gru_model.compile(optimizer='adam', loss='mean_squared_error')
gru_model.fit(X_train, y_train, epochs=20, batch_size=32, validation_split=0.2)

# Prepare data for ETS and ARIMA models
temp_max = data['temp_max'].values

# Fit ETS model
ets_model = ExponentialSmoothing(temp_max, seasonal='add', seasonal_periods=12)
ets_model_fit = ets_model.fit()
ets_predictions = ets_model_fit.forecast(steps=len(y_test))

# Fit ARIMA model
arima_model = ARIMA(temp_max, order=(5,1,0))
arima_model_fit = arima_model.fit()
arima_predictions = arima_model_fit.forecast(steps=len(y_test))
```

```
# Predict with LSTM and GRU
lstm_predictions = lstm_model.predict(X_test)
gru_predictions = gru_model.predict(X_test)

# Rescale predictions to original scale
lstm_predictions_rescaled = scaler.inverse_transform(np.concatenate([lstm_predictions, np.zeros((len(lstm_predictions), X.shape[2] - 1))], ax
gru_predictions_rescaled = scaler.inverse_transform(np.concatenate([gru_predictions, np.zeros((len(gru_predictions), X.shape[2] - 1))], axis=
y_test_rescaled = scaler.inverse_transform(np.concatenate([y_test.reshape(-1, 1), np.zeros((len(y_test), X.shape[2] - 1))], axis=1))[:, 0]

# Calculate mean squared error
mse_lstm = mean_squared_error(y_test_rescaled, lstm_predictions_rescaled)
mse_gru = mean_squared_error(y_test_rescaled, gru_predictions_rescaled)
mse_ets = mean_squared_error(temp_max[-len(ets_predictions):], ets_predictions)
mse_arima = mean_squared_error(temp_max[-len(arima_predictions):], arima_predictions)

print(f'MSE LSTM: {mse_lstm}')
print(f'MSE GRU: {mse_gru}')
print(f'MSE ETS: {mse_ets}')
print(f'MSE ARIMA: {mse_arima}')

# Plot the results
plt.figure(figsize=(12, 6))
plt.plot(data['date'][-len(y_test):], y_test_rescaled, label='Actual Values')
plt.plot(data['date'][-len(lstm_predictions):], lstm_predictions_rescaled, label='LSTM Predictions')
plt.plot(data['date'][-len(gru_predictions):], gru_predictions_rescaled, label='GRU Predictions')
plt.plot(data['date'][-len(ets_predictions):], ets_predictions, label='ETS Predictions')
plt.plot(data['date'][-len(arima_predictions):], arima_predictions, label='ARIMA Predictions')
plt.legend()
plt.xlabel('Date')
plt.ylabel('Temperature (Celsius)')
plt.title('Temperature Predictions')
plt.show()
```

```
29/29 [==============================] - 0s 7ms/step - loss: 0.0092 - val_loss: 0.012
Epoch 7/20
29/29 [==============================] - 0s 8ms/step - loss: 0.0085 - val_loss: 0.010
Epoch 8/20
29/29 [==============================] - 0s 7ms/step - loss: 0.0079 - val_loss: 0.008
Epoch 9/20
29/29 [==============================] - 0s 7ms/step - loss: 0.0076 - val_loss: 0.008
Epoch 10/20
29/29 [==============================] - 0s 7ms/step - loss: 0.0073 - val_loss: 0.008
Epoch 11/20
29/29 [==============================] - 0s 6ms/step - loss: 0.0071 - val_loss: 0.008
Epoch 12/20
29/29 [==============================] - 0s 8ms/step - loss: 0.0071 - val_loss: 0.007
Epoch 13/20
29/29 [==============================] - 0s 6ms/step - loss: 0.0068 - val_loss: 0.007
Epoch 14/20
29/29 [==============================] - 0s 7ms/step - loss: 0.0067 - val_loss: 0.006
Epoch 15/20
29/29 [==============================] - 0s 7ms/step - loss: 0.0067 - val_loss: 0.007
Epoch 16/20
29/29 [==============================] - 0s 6ms/step - loss: 0.0065 - val_loss: 0.006
Epoch 17/20
29/29 [==============================] - 0s 9ms/step - loss: 0.0067 - val_loss: 0.006
Epoch 18/20
29/29 [==============================] - 0s 6ms/step - loss: 0.0063 - val_loss: 0.006
Epoch 19/20
29/29 [==============================] - 0s 7ms/step - loss: 0.0063 - val_loss: 0.006
Epoch 20/20
29/29 [==============================] - 0s 6ms/step - loss: 0.0062 - val_loss: 0.006
10/10 [==============================] - 1s 5ms/step
10/10 [==============================] - 1s 4ms/step
MSE LSTM: 10.384549307942384
MSE GRU: 9.10139968855271
```

```
MSE ETS: 230.92739347777027
MSE ARIMA: 231.40273732903572
```




```
# Residuals for LSTM and GRU
lstm_residuals = y_test_rescaled - lstm_predictions_rescaled
gru_residuals = y_test_rescaled - gru_predictions_rescaled

plt.figure(figsize=(12, 6))
plt.subplot(2, 1, 1)
plt.plot(data['date'][-len(y_test):], lstm_residuals, label='LSTM Residuals')
plt.title('LSTM Residuals')
plt.xlabel('Date')
plt.ylabel('Residuals')
plt.legend()

plt.subplot(2, 1, 2)
plt.plot(data['date'][-len(y_test):], gru_residuals, label='GRU Residuals')
plt.title('GRU Residuals')
plt.xlabel('Date')
plt.ylabel('Residuals')
plt.legend()

plt.tight_layout()
plt.show()
```