[3]

```python
import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder
from sklearn.feature_selection import SelectKBest, chi2, mutual_info_regression, f_regression, RFE
from sklearn.ensemble import RandomForestRegressor
from sklearn.linear_model import LinearRegression
df = pd.read_csv("CGWB_Groundwater_Level.csv")
```

[4]
✓ 0s

```python
# ----------------- FEATURE GENERATION -----------------

# convert date column to datetime
df['date'] = pd.to_datetime(df['date'])

# Temporal features
df['year'] = df['date'].dt.year
df['month'] = df['date'].dt.month
df['day'] = df['date'].dt.day
df['day_of_week'] = df['date'].dt.dayofweek

# Mathematical features
df['abs_diff'] = df['level_diff'].abs()
df['square_level'] = df['currentlevel'] ** 2
df['log_level'] = np.log1p(df['currentlevel'])

# Encoding categorical columns
encoder = LabelEncoder()
df['state_enc'] = encoder.fit_transform(df['state_name'])
df['district_enc'] = encoder.fit_transform(df['district_name'])
df['basin_enc'] = encoder.fit_transform(df['basin'])

# Interaction features
df['level_ratio'] = df['currentlevel'] / (df['level_diff'] + 1)

# Text features
df['basin_word_count'] = df['basin'].apply(lambda x: len(str(x).split()))
```

[16]
✓ 0s

```python
generated_cols = [
    'year', 'month', 'day', 'day_of_week',
    'abs_diff', 'square_level', 'log_level',
    'state_enc', 'district_enc', 'basin_enc',
    'level_ratio', 'basin_word_count'
]

print("Generated Columns:\n", df[generated_cols].head())
```

```
Generated Columns:
    year  month  day  day_of_week  abs_diff  square_level  log_level  \
0  2013     11    4            0      1.03        0.0100   0.095310
1  2014      5   14            2      2.50        6.7600   1.280934
2  2014     11    4            1      2.25        0.1225   0.300105
3  2015      5   14            3      2.17        6.3504   1.258461
4  2015     11    4            2      1.83        0.4761   0.524729

   state_enc  district_enc  basin_enc  level_ratio  basin_word_count
0          0            40          2    -3.333333                 8
1          0            40          2     0.742857                 8
2          0            40          2    -0.280000                 8
3          0            40          2     0.794953                 8
4          0            40          2    -0.831325                 8
```

[15]
✓ 1m

```python
import pandas as pd
import numpy as np
from sklearn.feature_selection import SelectKBest, chi2, mutual_info_regression, f_regression, RFE
from sklearn.ensemble import RandomForestRegressor
from sklearn.linear_model import LinearRegression

# take numeric columns
num_df = df.select_dtypes(include=['int64', 'float64'])

# drop rows with missing target
num_df = num_df.dropna(subset=['currentlevel'])

X = num_df.drop(columns=['currentlevel'])
y = num_df['currentlevel']

# replace inf/-inf and fill missing values
X = X.replace([np.inf, -np.inf], np.nan)
X = X.fillna(X.median())
y = y.fillna(y.median())

# 1. Correlation
corr = X.corrwith(y).abs().sort_values(ascending=False)
print("Top correlated features:\n", corr.head(5))
```

[15]
✓ 1m

```python
# 2. Chi-Square test (use categorical y)
y_chi = pd.qcut(y, q=3, labels=[0, 1, 2])  # convert y to 3 categories
X_chi = X.apply(lambda x: x - x.min())     # make non-negative
chi_selector = SelectKBest(score_func=chi2, k=5)
chi_selector.fit(X_chi, y_chi)
print("\nChi-square top features:", list(X.columns[chi_selector.get_support()]))

# 3. Mutual Information
mi_selector = SelectKBest(score_func=mutual_info_regression, k=5)
mi_selector.fit(X, y)
print("\nMutual Information top features:", list(X.columns[mi_selector.get_support()]))

# 4. ANOVA test
anova_selector = SelectKBest(score_func=f_regression, k=5)
anova_selector.fit(X, y)
print("\nANOVA top features:", list(X.columns[anova_selector.get_support()]))

# 5. Wrapper method (RFE)
rfe = RFE(LinearRegression(), n_features_to_select=5)
rfe.fit(X, y)
print("\nRFE top features:", list(X.columns[rfe.support_]))

# 6. Embedded method (Random Forest)
rf = RandomForestRegressor(random_state=42)
rf.fit(X, y)
importance = pd.Series(rf.feature_importances_, index=X.columns)
print("\nRandom Forest important features:\n", importance.nlargest(5))
```

```
Top correlated features:
 square_level    0.957650
log_level       0.933365
state_enc       0.372792
id              0.348304
level_diff      0.256168
dtype: float64

Chi-square top features: ['id', 'district_code', 'square_level', 'log_level', 'basin_word_count']

Mutual Information top features: ['id', 'level_diff', 'square_level', 'log_level', 'level_ratio']

ANOVA top features: ['id', 'level_diff', 'square_level', 'log_level', 'state_enc']

RFE top features: ['state_code', 'latitude', 'longitude', 'log_level', 'state_enc']

Random Forest important features:
 square_level    5.367166e-01
log_level       4.632795e-01
level_diff      7.095484e-07
abs_diff        6.534364e-07
district_enc    6.370616e-07
dtype: float64
```