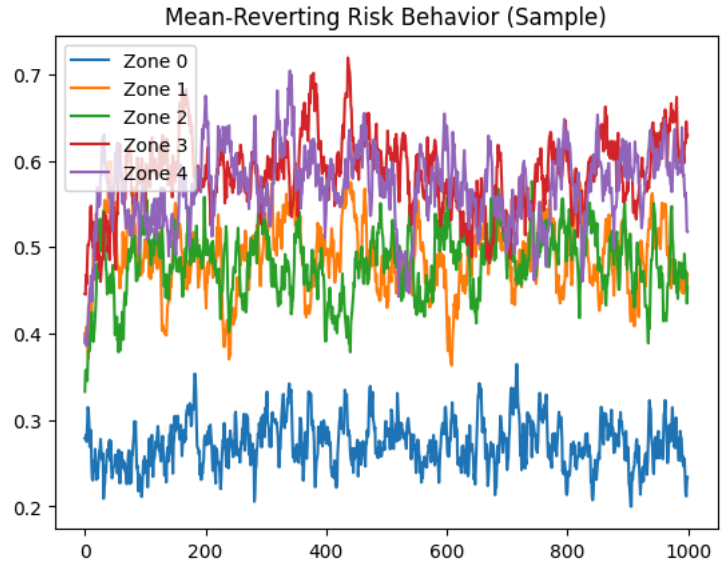
```
import pandas as pd
df = pd.read_csv("rockfall_mean_reverting_professional_dataset_v3_50k.csv")

df.groupby("zone_id")["risk_index"].describe()
```

zone_id	count	mean	std	min	25%	50%	75%	max
0	10000.0	0.272298	0.031508	0.160280	0.250737	0.272027	0.294143	0.387757
1	10000.0	0.489146	0.036588	0.348476	0.465635	0.490119	0.513642	0.642818
2	10000.0	0.476515	0.035993	0.332817	0.453023	0.477125	0.501277	0.603594
3	10000.0	0.588899	0.038798	0.445036	0.562873	0.588986	0.614753	0.718999
4	10000.0	0.570920	0.040204	0.385553	0.543647	0.570370	0.597466	0.716883

```
import matplotlib.pyplot as plt

for z in range(5):
    zone = df[df["zone_id"] == z]
    plt.plot(zone["risk_index"].values[:1000], label=f"Zone {z}")

plt.legend()
plt.title("Mean-Reverting Risk Behavior (Sample)")
plt.show()
```




```
import numpy as np
from sklearn.preprocessing import MinMaxScaler

FEATURES = [
    "displacement",
    "strain",
    "pore_pressure",
    "vibration",
    "rainfall",
    "temperature"
]

TARGET = "risk_index"

X = df[FEATURES]
y = df[TARGET]

scaler = MinMaxScaler()
X_scaled = scaler.fit_transform(X)
```

```
WINDOW = 30

X_seq, y_seq, zone_seq = [], [], []
```

```
for zone_id in df["zone_id"].unique():
    mask = df["zone_id"] == zone_id
    X_zone = X_scaled[mask]
    y_zone = y[mask].values

    for i in range(len(X_zone) - WINDOW):
        X_seq.append(X_zone[i:i+WINDOW])
        y_seq.append(y_zone[i+WINDOW])
        zone_seq.append(zone_id)

X_seq = np.array(X_seq)
y_seq = np.array(y_seq)
zone_seq = np.array(zone_seq)

print(X_seq.shape, y_seq.shape, np.unique(zone_seq))
```

```
(49850, 30, 6) (49850,) [0 1 2 3 4]
```

```
X_train, X_test = [], []
y_train, y_test = [], []
zone_test = []

for z in np.unique(zone_seq):
    idx = np.where(zone_seq == z)[0]
    split = int(0.8 * len(idx))

    X_train.append(X_seq[idx[:split]])
    X_test.append(X_seq[idx[split:]])

    y_train.append(y_seq[idx[:split]])
    y_test.append(y_seq[idx[split:]])

    zone_test.extend([z] * (len(idx) - split))

X_train = np.concatenate(X_train)
X_test = np.concatenate(X_test)
y_train = np.concatenate(y_train)
y_test = np.concatenate(y_test)
zone_test = np.array(zone_test)

print("Train:", X_train.shape)
print("Test:", X_test.shape)
print("Zones in test:", np.unique(zone_test))
```

```
Train: (39880, 30, 6)
Test: (9970, 30, 6)
Zones in test: [0 1 2 3 4]
```

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout
from tensorflow.keras.callbacks import EarlyStopping
```

```
model = Sequential([
    LSTM(64, input_shape=(30, 6)),
    Dropout(0.2),
    Dense(1)
])

model.compile(
    optimizer="adam",
    loss="mae",
    metrics=[tf.keras.metrics.RootMeanSquaredError()]
)

model.summary()
```

```
/usr/local/lib/python3.12/dist-packages/keras/src/layers/rnn/rnn.py:199: UserWarning: Do not pass an `input_shape
  super().__init__(**kwargs)
```

Model: "sequential"

Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 64)	18,176
dropout (Dropout)	(None, 64)	0
dense (Dense)	(None, 1)	65

Total params: 18,241 (71.25 KB)
Trainable params: 18,241 (71.25 KB)
Non-trainable params: 0 (0.00 B)

```
early_stop = EarlyStopping(
    monitor="val_loss",
    patience=5,
    restore_best_weights=True
)

history = model.fit(
    X_train,
    y_train,
    validation_split=0.2,
    epochs=40,
    batch_size=64,
    callbacks=[early_stop],
    verbose=1
)
```

```
499/499 ━━━━━━━━━━━━━━━━━━━━ 21s 23ms/step - loss: 0.0312 - root_mean_squared_error: 0.0394 - val_loss: 0.0230 -
Epoch 4/40
499/499 ━━━━━━━━━━━━━━━━━━━━ 12s 23ms/step - loss: 0.0284 - root_mean_squared_error: 0.0358 - val_loss: 0.0228 -
Epoch 5/40
499/499 ━━━━━━━━━━━━━━━━━━━━ 20s 21ms/step - loss: 0.0273 - root_mean_squared_error: 0.0345 - val_loss: 0.0233 -
Epoch 6/40
499/499 ━━━━━━━━━━━━━━━━━━━━ 12s 23ms/step - loss: 0.0257 - root_mean_squared_error: 0.0324 - val_loss: 0.0217 -
Epoch 7/40
499/499 ━━━━━━━━━━━━━━━━━━━━ 11s 23ms/step - loss: 0.0248 - root_mean_squared_error: 0.0312 - val_loss: 0.0271 -
Epoch 8/40
499/499 ━━━━━━━━━━━━━━━━━━━━ 12s 23ms/step - loss: 0.0233 - root_mean_squared_error: 0.0294 - val_loss: 0.0221 -
Epoch 9/40
499/499 ━━━━━━━━━━━━━━━━━━━━ 11s 23ms/step - loss: 0.0227 - root_mean_squared_error: 0.0286 - val_loss: 0.0202 -
Epoch 10/40
499/499 ━━━━━━━━━━━━━━━━━━━━ 20s 23ms/step - loss: 0.0222 - root_mean_squared_error: 0.0279 - val_loss: 0.0200 -
Epoch 11/40
499/499 ━━━━━━━━━━━━━━━━━━━━ 12s 23ms/step - loss: 0.0218 - root_mean_squared_error: 0.0274 - val_loss: 0.0209 -
Epoch 12/40
499/499 ━━━━━━━━━━━━━━━━━━━━ 11s 23ms/step - loss: 0.0215 - root_mean_squared_error: 0.0271 - val_loss: 0.0216 -
Epoch 13/40
499/499 ━━━━━━━━━━━━━━━━━━━━ 12s 23ms/step - loss: 0.0214 - root_mean_squared_error: 0.0269 - val_loss: 0.0197 -
Epoch 14/40
499/499 ━━━━━━━━━━━━━━━━━━━━ 19s 20ms/step - loss: 0.0210 - root_mean_squared_error: 0.0265 - val_loss: 0.0207 -
Epoch 15/40
499/499 ━━━━━━━━━━━━━━━━━━━━ 11s 22ms/step - loss: 0.0211 - root_mean_squared_error: 0.0267 - val_loss: 0.0203 -
Epoch 16/40
499/499 ━━━━━━━━━━━━━━━━━━━━ 11s 22ms/step - loss: 0.0211 - root_mean_squared_error: 0.0264 - val_loss: 0.0196 -
Epoch 17/40
499/499 ━━━━━━━━━━━━━━━━━━━━ 22s 25ms/step - loss: 0.0210 - root_mean_squared_error: 0.0264 - val_loss: 0.0195 -
Epoch 18/40
499/499 ━━━━━━━━━━━━━━━━━━━━ 11s 21ms/step - loss: 0.0207 - root_mean_squared_error: 0.0261 - val_loss: 0.0226 -
Epoch 19/40
499/499 ━━━━━━━━━━━━━━━━━━━━ 21s 23ms/step - loss: 0.0207 - root_mean_squared_error: 0.0260 - val_loss: 0.0197 -
Epoch 20/40
499/499 ━━━━━━━━━━━━━━━━━━━━ 20s 23ms/step - loss: 0.0207 - root_mean_squared_error: 0.0260 - val_loss: 0.0212 -
Epoch 21/40
499/499 ━━━━━━━━━━━━━━━━━━━━ 19s 20ms/step - loss: 0.0206 - root_mean_squared_error: 0.0259 - val_loss: 0.0257 -
Epoch 22/40
499/499 ━━━━━━━━━━━━━━━━━━━━ 11s 23ms/step - loss: 0.0211 - root_mean_squared_error: 0.0267 - val_loss: 0.0192 -
Epoch 23/40
499/499 ━━━━━━━━━━━━━━━━━━━━ 11s 22ms/step - loss: 0.0205 - root_mean_squared_error: 0.0258 - val_loss: 0.0194 -
Epoch 24/40
499/499 ━━━━━━━━━━━━━━━━━━━━ 11s 22ms/step - loss: 0.0203 - root_mean_squared_error: 0.0256 - val_loss: 0.0192 -
Epoch 25/40
499/499 ━━━━━━━━━━━━━━━━━━━━ 19s 20ms/step - loss: 0.0205 - root_mean_squared_error: 0.0257 - val_loss: 0.0192 -
Epoch 26/40
499/499 ━━━━━━━━━━━━━━━━━━━━ 12s 23ms/step - loss: 0.0205 - root_mean_squared_error: 0.0257 - val_loss: 0.0192 -
Epoch 27/40
499/499 ━━━━━━━━━━━━━━━━━━━━ 11s 23ms/step - loss: 0.0206 - root_mean_squared_error: 0.0259 - val_loss: 0.0211 -
```

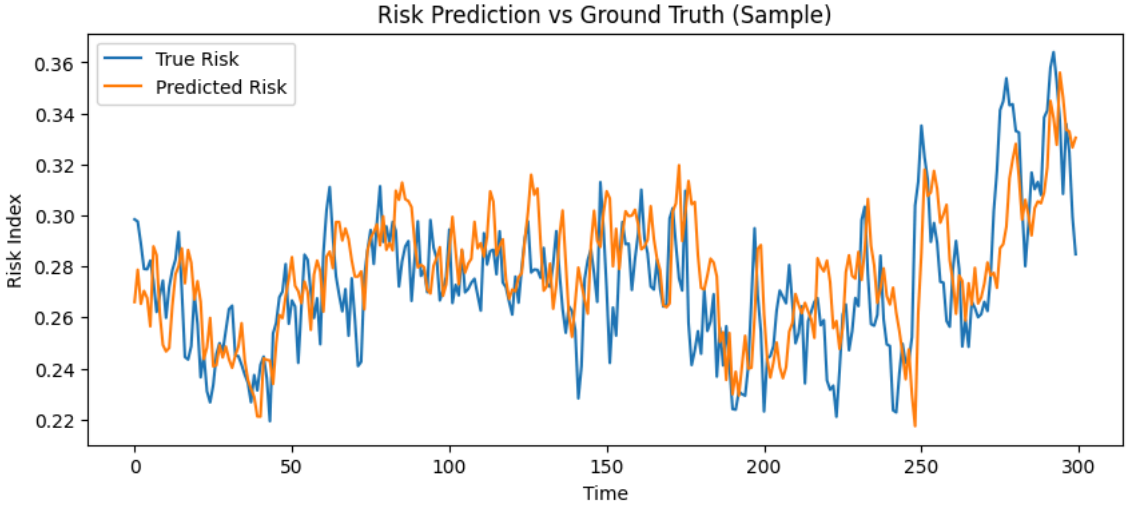
```
499/499 ━━━━━━━━━━━━━━━━━━━━ 10s 20ms/step - loss: 0.0203 - root_mean_squared_error: 0.0255 - val_loss: 0.0195 -
Epoch 31/40
499/499 ━━━━━━━━━━━━━━━━━━━━ 11s 22ms/step - loss: 0.0204 - root mean squared error: 0.0256 - val loss: 0.0194 -
```

```
test_loss, test_rmse = model.evaluate(X_test, y_test, verbose=0)

print(f"Test MAE: {test_loss:.4f}")
print(f"Test RMSE: {test_rmse:.4f}")
```

```
Test MAE: 0.0189
Test RMSE: 0.0239
```

```
import matplotlib.pyplot as plt

y_pred = model.predict(X_test).flatten()

plt.figure(figsize=(10,4))
plt.plot(y_test[:300], label="True Risk")
plt.plot(y_pred[:300], label="Predicted Risk")
plt.legend()
plt.title("Risk Prediction vs Ground Truth (Sample)")
plt.xlabel("Time")
plt.ylabel("Risk Index")
plt.show()
```

```
312/312 ━━━━━━━━━━━━━━━━━━━━ 2s 5ms/step
```




```
y_pred = model.predict(X_test).flatten()

import pandas as pd

pred_df = pd.DataFrame({
    "zone_id": zone_test,
    "true_risk": y_test,
    "predicted_risk": y_pred
})

pred_df.head()
```

```
312/312 ━━━━━━━━━━━━━━━━━━━━ 2s 5ms/step
```

	zone_id	true_risk	predicted_risk
0	0	0.298462	0.266043
1	0	0.297595	0.278684
2	0	0.288908	0.265322
3	0	0.279042	0.270405
4	0	0.278902	0.267602

Next steps: Generate code with pred_df | New interactive sheet

```
zone_baseline = (
    pred_df
    .groupby("zone_id")["predicted_risk"]
    .median()
```

```
    .to_dict()
)

zone_baseline
```

```
{0: 0.28040611743927,
 1: 0.4919623136520386,
 2: 0.4929134249687195,
 3: 0.5850605964660645,
 4: 0.5711876153945923}
```

```
WINDOW_ALERT = 10

pred_df["rolling_risk"] = (
    pred_df
    .groupby("zone_id")["predicted_risk"]
    .rolling(WINDOW_ALERT)
    .mean()
    .reset_index(level=0, drop=True)
)
```

```
import numpy as np

def rolling_slope(series):
    if len(series) < 2:
        return 0.0
    x = np.arange(len(series))
    return np.polyfit(x, series, 1)[0]

pred_df["risk_trend"] = (
    pred_df
    .groupby("zone_id")["predicted_risk"]
    .rolling(WINDOW_ALERT)
    .apply(rolling_slope, raw=False)
    .reset_index(level=0, drop=True)
)
```

```
pred_df["baseline_deviation"] = pred_df.apply(
    lambda r: r["rolling_risk"] - zone_baseline[r["zone_id"]],
    axis=1
)
```

```
def compute_risk_level(row):
    if pd.isna(row["rolling_risk"]):
        return "INSUFFICIENT DATA"
    if row["baseline_deviation"] > 0.08 and row["risk_trend"] > 0:
        return "ALERT"
    elif row["baseline_deviation"] > 0.04:
        return "WATCH"
    else:
        return "SAFE"

pred_df["risk_level"] = pred_df.apply(compute_risk_level, axis=1)

pred_df[["zone_id", "rolling_risk", "risk_trend", "risk_level"]].tail(20)
```

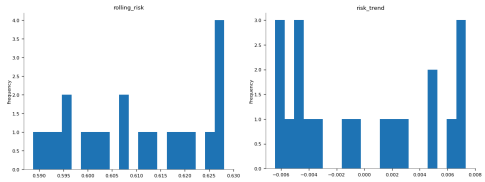
	zone_id	rolling_risk	risk_trend	risk_level
9950	4	0.592736	0.001431	SAFE

```
ALERT_PERSISTENCE = 3

pred_df["alert_flag"] = pred_df["risk_level"] == "ALERT"

pred_df["persistent_alert"] = (
    pred_df
    .groupby("zone_id")["alert_flag"]
    .rolling(ALERT_PERSISTENCE)
    .sum()
    .reset_index(level=0, drop=True)
    >= ALERT_PERSISTENCE
)
```

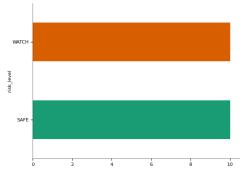
```
def alert_message(row):
    if row["persistent_alert"]:
        return "🚨 HIGH RISK: Immediate inspection required."
    elif row["risk_level"] == "WATCH":
        return "⚠️ WATCH: Risk increasing, monitor closely."
    else:
        return "✅ SAFE: Normal operating conditions."

pred_df["alert_message"] = pred_df.apply(alert_message, axis=1)

pred_df[["zone_id", "risk_level", "persistent_alert", "alert_message"]].tail(20)
```

9968	4	0.591459	-0.004605	SAFE
9969	4	0.588784	-0.003981	SAFE

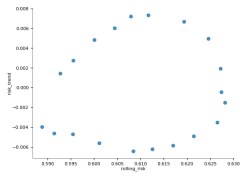
Distributions



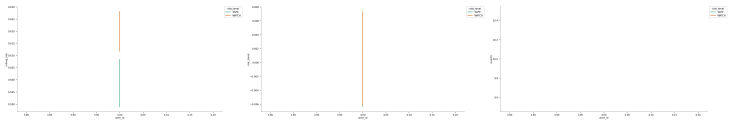
Categorical distributions



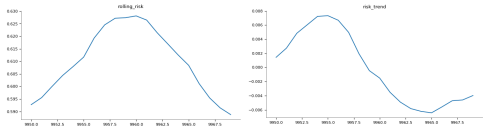
2-d distributions



Time series

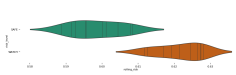


Values



Faceted distributions

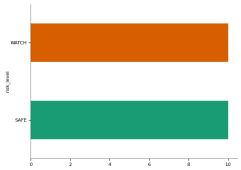
```
<string>:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable
```



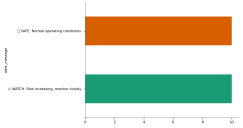
```
<string>:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable
```

	zone_id	risk_level	persistent_alert	alert_message
9950	4	SAFE	False	✅ SAFE: Normal operating conditions.
9951	4	SAFE	False	✅ SAFE: Normal operating conditions.
9952	4	SAFE	False	✅ SAFE: Normal operating conditions.
9953	4	SAFE	False	✅ SAFE: Normal operating conditions.
9954	4	SAFE	False	✅ SAFE: Normal operating conditions.
9955	4	WATCH	False	⚠️ WATCH: Risk increasing, monitor closely.
9956	4	WATCH	False	⚠️ WATCH: Risk increasing, monitor closely.
9957	4	WATCH	False	⚠️ WATCH: Risk increasing, monitor closely.
9958	4	WATCH	False	⚠️ WATCH: Risk increasing, monitor closely.
9959	4	WATCH	False	⚠️ WATCH: Risk increasing, monitor closely.
9960	4	WATCH	False	⚠️ WATCH: Risk increasing, monitor closely.
9961	4	WATCH	False	⚠️ WATCH: Risk increasing, monitor closely.
9962	4	WATCH	False	⚠️ WATCH: Risk increasing, monitor closely.
9963	4	WATCH	False	⚠️ WATCH: Risk increasing, monitor closely.
9964	4	WATCH	False	⚠️ WATCH: Risk increasing, monitor closely.
9965	4	SAFE	False	✅ SAFE: Normal operating conditions.
9966	4	SAFE	False	✅ SAFE: Normal operating conditions.
9967	4	SAFE	False	✅ SAFE: Normal operating conditions.
9968	4	SAFE	False	✅ SAFE: Normal operating conditions.
9969	4	SAFE	False	✅ SAFE: Normal operating conditions.

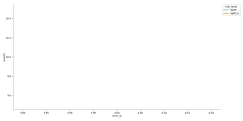
Categorical distributions



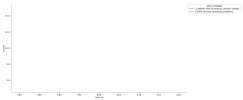

```
/usr/local/lib/python3.12/dist-packages/google/colab/_quickchart_lib.py:32: UserWarning: Glyph 9989 (\N{WHITE HI
  plt.savefig(
```



Time series

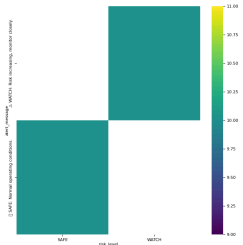


```
/usr/local/lib/python3.12/dist-packages/google/colab/_quickchart_lib.py:32: UserWarning: Glyph 9989 (\N{WHITE HI
  plt.savefig(
```



2-d categorical distributions

```
/usr/local/lib/python3.12/dist-packages/seaborn/utils.py:61: UserWarning: Glyph 9989 (\N{WHITE HEAVY CHECK MARK
  fig.canvas.draw()
/usr/local/lib/python3.12/dist-packages/google/colab/_quickchart_lib.py:32: UserWarning: Glyph 9989 (\N{WHITE HI
  plt.savefig(
```




```
pred_df["zone_id"].value_counts()
```

zone_id	count
0	1994
1	1994
2	1994
3	1994
4	1994

dtype: int64

```
pd.crosstab(pred_df["zone_id"], pred_df["risk_level"])
```

zone_id	ALERT	INSUFFICIENT DATA	SAFE	WATCH
0	0	9	1922	63
1	0	9	1876	109
2	0	9	1931	54
3	4	9	1766	215
4	0	9	1815	170

```
pred_df.sample(20).sort_values("zone_id")
```

	zone_id	true_risk	predicted_risk	rolling_risk	risk_trend	baseline_deviation	risk_level	alert_flag
1325	0	0.242393	0.226496	0.261001	-0.005336	-0.019405	SAFE	False
1949	0	0.270319	0.247042	0.256251	-0.000231	-0.024155	SAFE	False
471	0	0.309771	0.345000	0.318835	0.005916	0.038428	SAFE	False
386	0	0.299631	0.289301	0.277060	0.001272	-0.003346	SAFE	False
3592	1	0.479799	0.437061	0.447820	-0.007268	-0.044142	SAFE	False
2085	1	0.461560	0.414575	0.432032	-0.002711	-0.059930	SAFE	False
3759	1	0.464594	0.485945	0.512717	-0.005823	0.020755	SAFE	False
2178	1	0.522202	0.529458	0.515567	0.000799	0.023605	SAFE	False
3192	1	0.467712	0.492240	0.478408	-0.000726	-0.013554	SAFE	False
5036	2	0.430076	0.437825	0.494069	-0.009497	0.001155	SAFE	False
5222	2	0.456652	0.485117	0.476849	0.000193	-0.016064	SAFE	False
5285	2	0.472963	0.485495	0.486299	0.003536	-0.006615	SAFE	False
4072	2	0.496077	0.500005	0.489940	-0.003258	-0.002974	SAFE	False
5983	3	0.484584	0.567726	NaN	NaN	NaN	INSUFFICIENT DATA	False
6077	3	0.531177	0.549355	0.584297	-0.005379	-0.000764	SAFE	False

COMPARISON MACHINE LEARNING MODEL

```
# Flatten sequence windows
X_train_flat = X_train.reshape(X_train.shape[0], -1)
X_test_flat  = X_test.reshape(X_test.shape[0], -1)

print(X_train_flat.shape, X_test_flat.shape)
```

```
(39880, 180) (9970, 180)
```

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error
import numpy as np

lr = LinearRegression()
lr.fit(X_train_flat, y_train)

y_pred_lr = lr.predict(X_test_flat)
```

```
from sklearn.ensemble import RandomForestRegressor

rf = RandomForestRegressor(
    n_estimators=100,
```

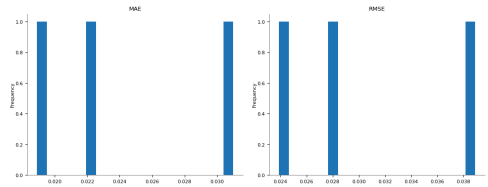
```
    max_depth=10,
    random_state=42,
    n_jobs=-1
)

rf.fit(X_train_flat, y_train)
y_pred_rf = rf.predict(X_test_flat)
```

```
def evaluate_model(name, y_true, y_pred):
    mae = mean_absolute_error(y_true, y_pred)
    rmse = np.sqrt(mean_squared_error(y_true, y_pred))
    return {
        "Model": name,
        "MAE": round(mae, 4),
        "RMSE": round(rmse, 4)
    }
```

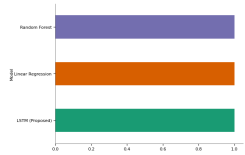
```
results = []

results.append(evaluate_model("Linear Regression", y_test, y_pred_lr))
results.append(evaluate_model("Random Forest", y_test, y_pred_rf))
results.append(evaluate_model("LSTM (Proposed)", y_test, y_pred))

import pandas as pd
results_df = pd.DataFrame(results)
results_df
```

	Model	MAE	RMSE
0	Linear Regression	0.0221	0.0278
1	Random Forest	0.0310	0.0389
2	LSTM (Proposed)	0.0189	0.0239

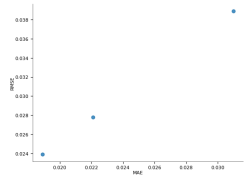
Distributions



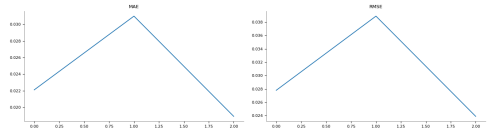
Categorical distributions



2-d distributions

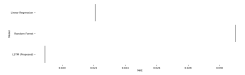


Values

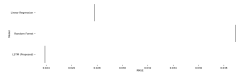


Faceted distributions

```
<string>:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable t
```



```
<string>:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable t
```



Next steps: Generate code with `results_df` New interactive sheet

```
results_df.to_csv("model_comparison_results.csv", index=False)
```

```
!pip install -q gradio matplotlib seaborn
```

```
import gradio as gr
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

```
# Must already be computed earlier
# pred_df columns required:
# ['zone_id','rolling_risk','risk_level','alert_message']

pred_df.head()
```