Predicting Rainfall with Machine Learning

By Lynne, Naresh, Deepak, Mridul

DS207 Spring 2025

April 17 2025

[Dataset: Kaggle · Playground Prediction Competition]

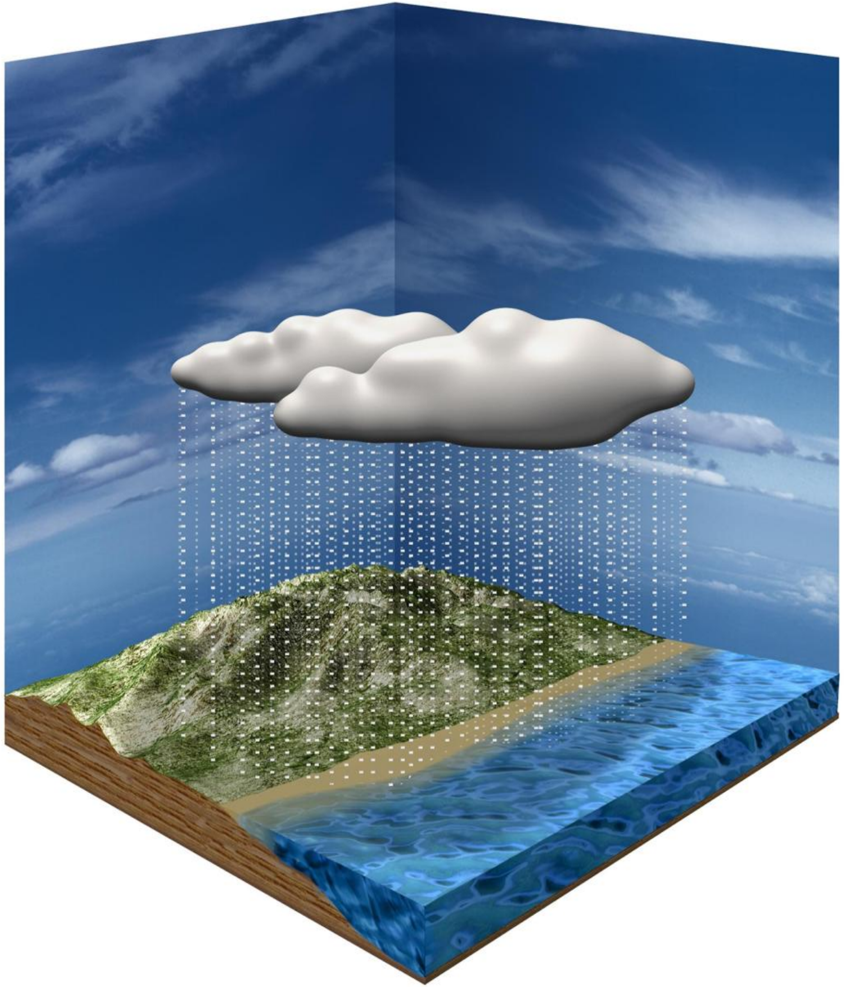
[https://github.com/lwang9/mids-w207-final__project__team2]

Berkeley
UNIVERSITY OF CALIFORNIA

Motivation

- ➢ Question: Will it rain on any given day?

- ➢ Why is it interesting? Real-World Impact:
  - ▪ Agriculture.
  - ▪ Disaster preparedness.
  - ▪ Urban planning.
  - ▪ Commute/travel planning.
  - ▪ ….



Previous Forecasting Methods

- Numerical Weather Prediction (NWP) Models.
- Statistical Models.
- Satellite-Based Models.

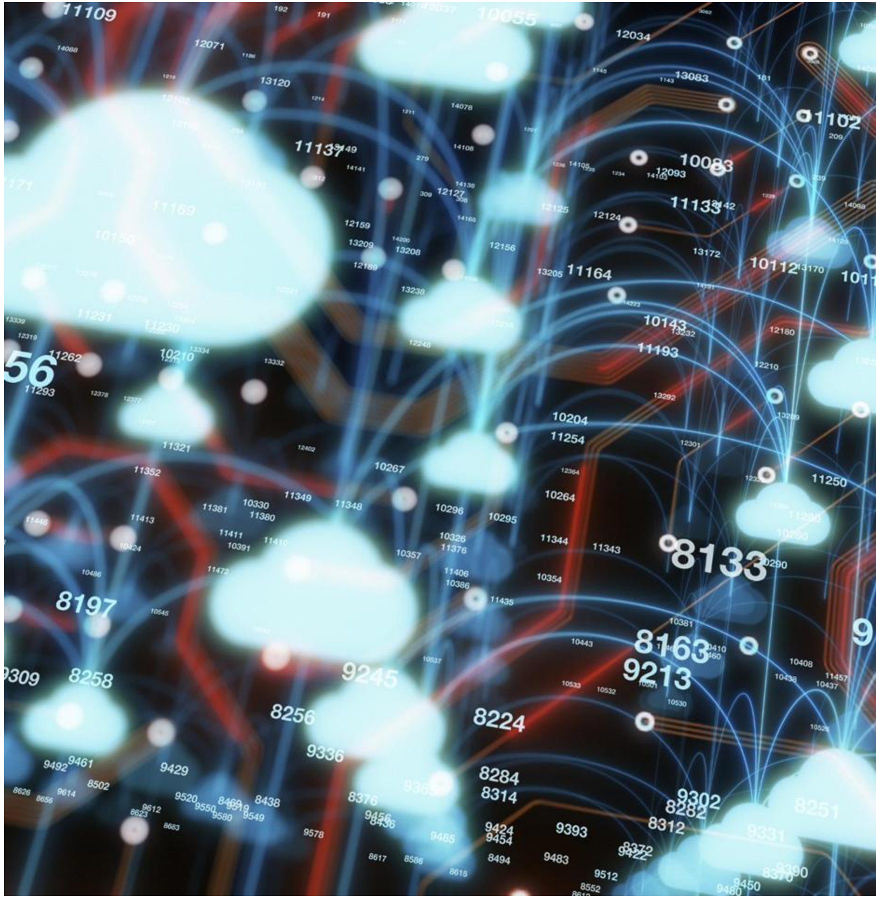
https://www.pivotalweather.com

Berkeley
UNIVERSITY OF CALIFORNIA

Binary Classification

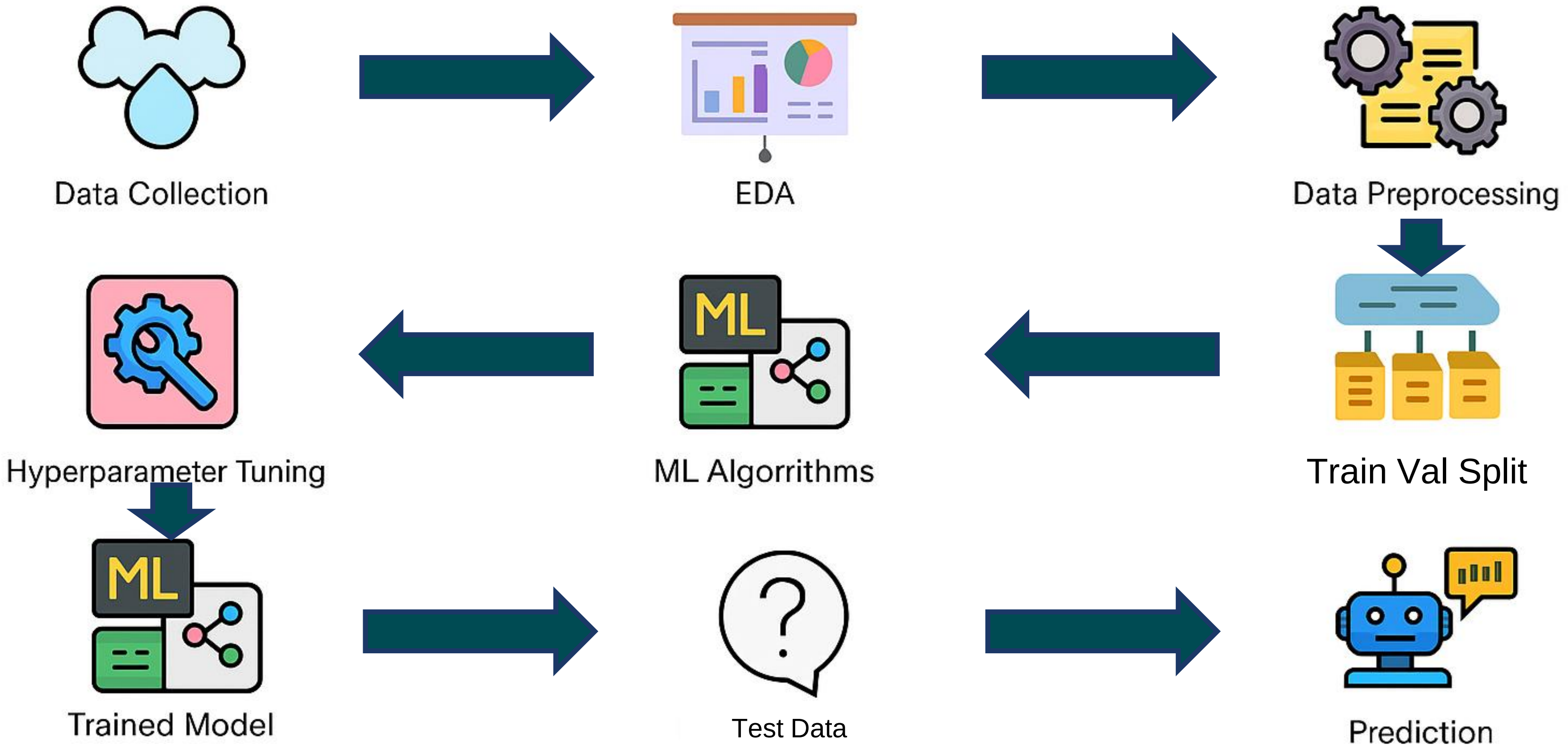
- ➢ Given Input Features:
  ID, day, pressure, maxtemp, temperature, mintemp, dewpoint, humidity, cloud, sunshine, wind direction, windspeed (2190 rows)

- ➢ Output: rainfall
  1 = rain, 0 = no rain



Reference: Generated by chatgpt

Overall Plan



Summary of Findings

- Many models perform reasonably well.
- Ensemble modeling achieved near-top leaderboard performance on Kaggle (rank 21/4382).

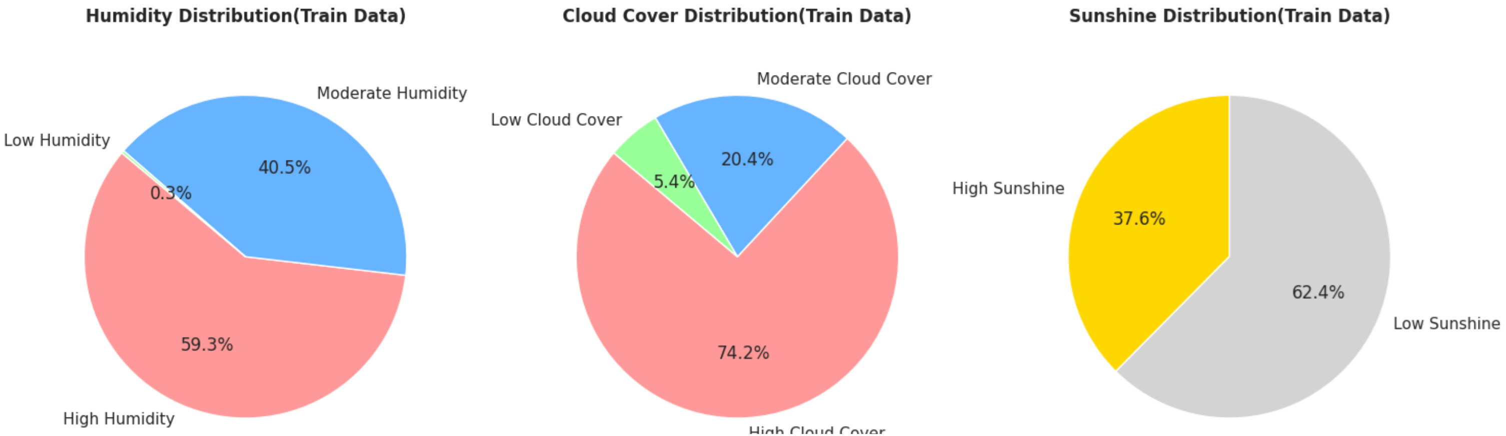
#	△	Team	Members	Score
1	▲ 812	Guillaume HIMBERT		0.90654
21	▲ 2242	AvasthiPrakhar		0.90376

submission_ensemble.csv **0.90376**

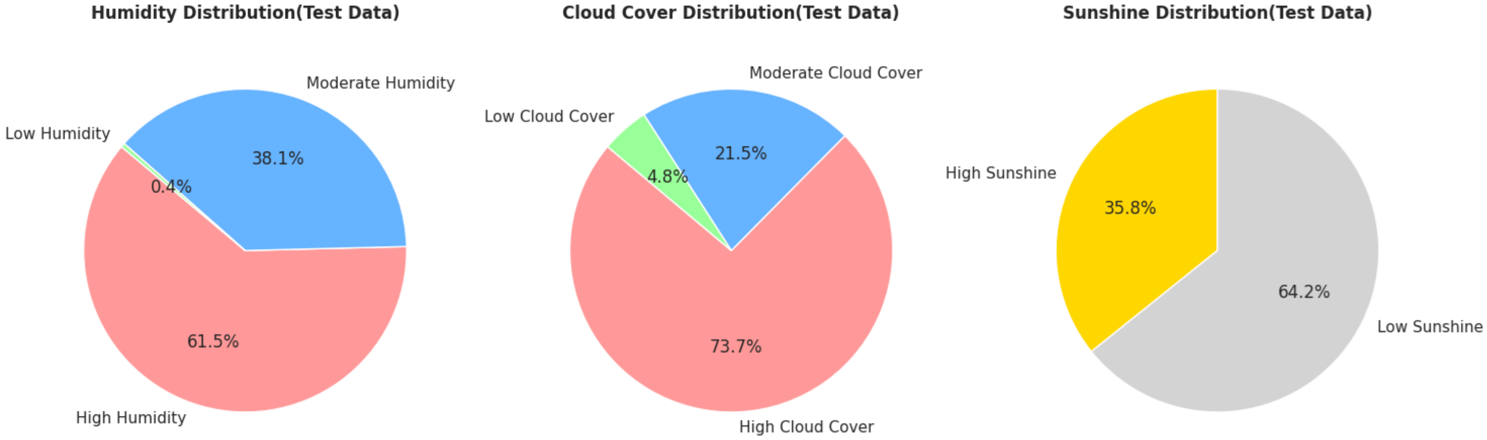
Complete (after deadline) · Lynne Wang · 7d ago

EDA - Comparison of key variables (train & test)

Train Data

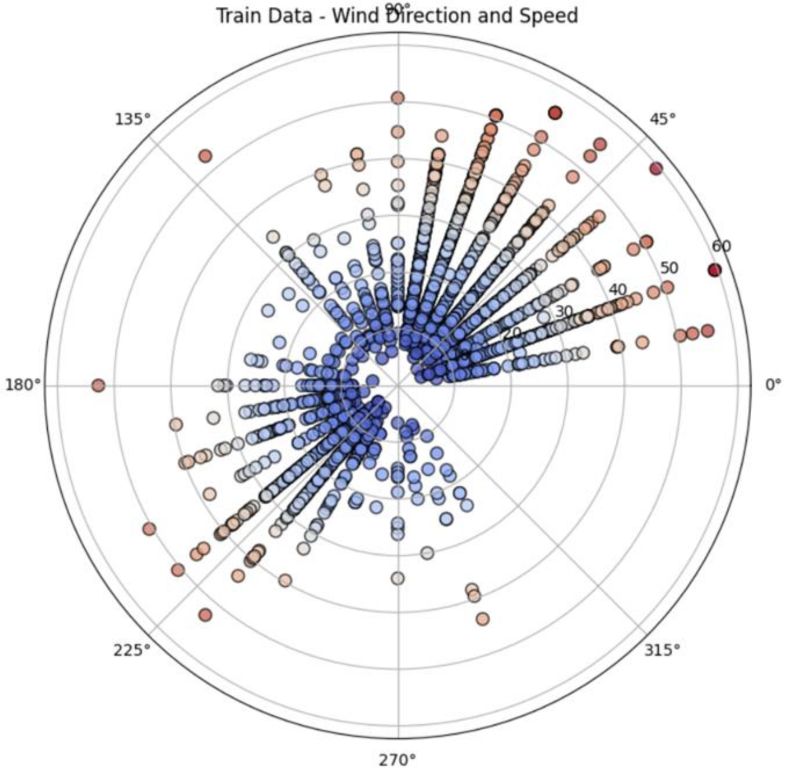


Test Data

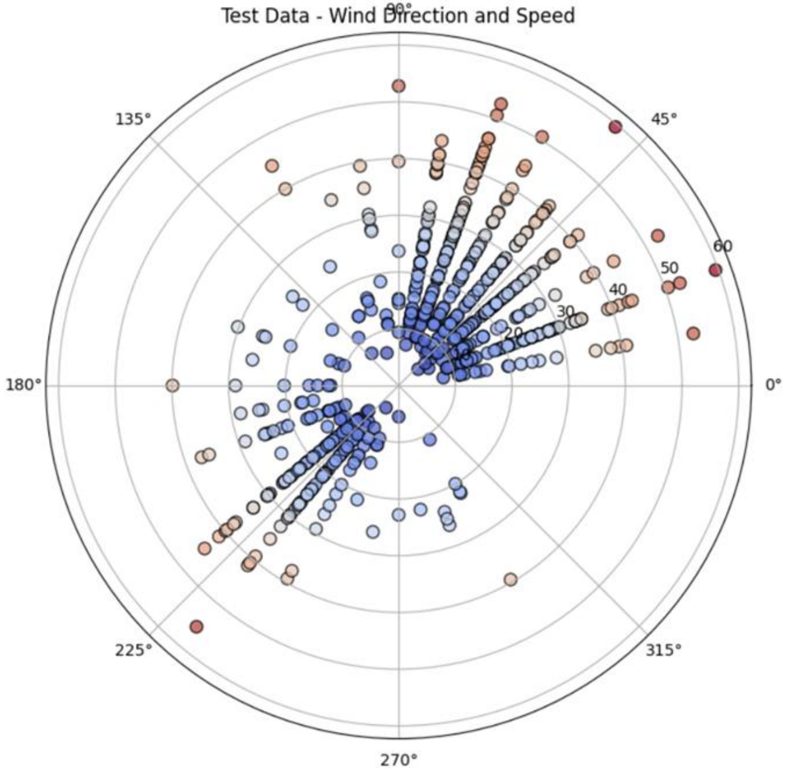


EDA - Wind Direction & Speed

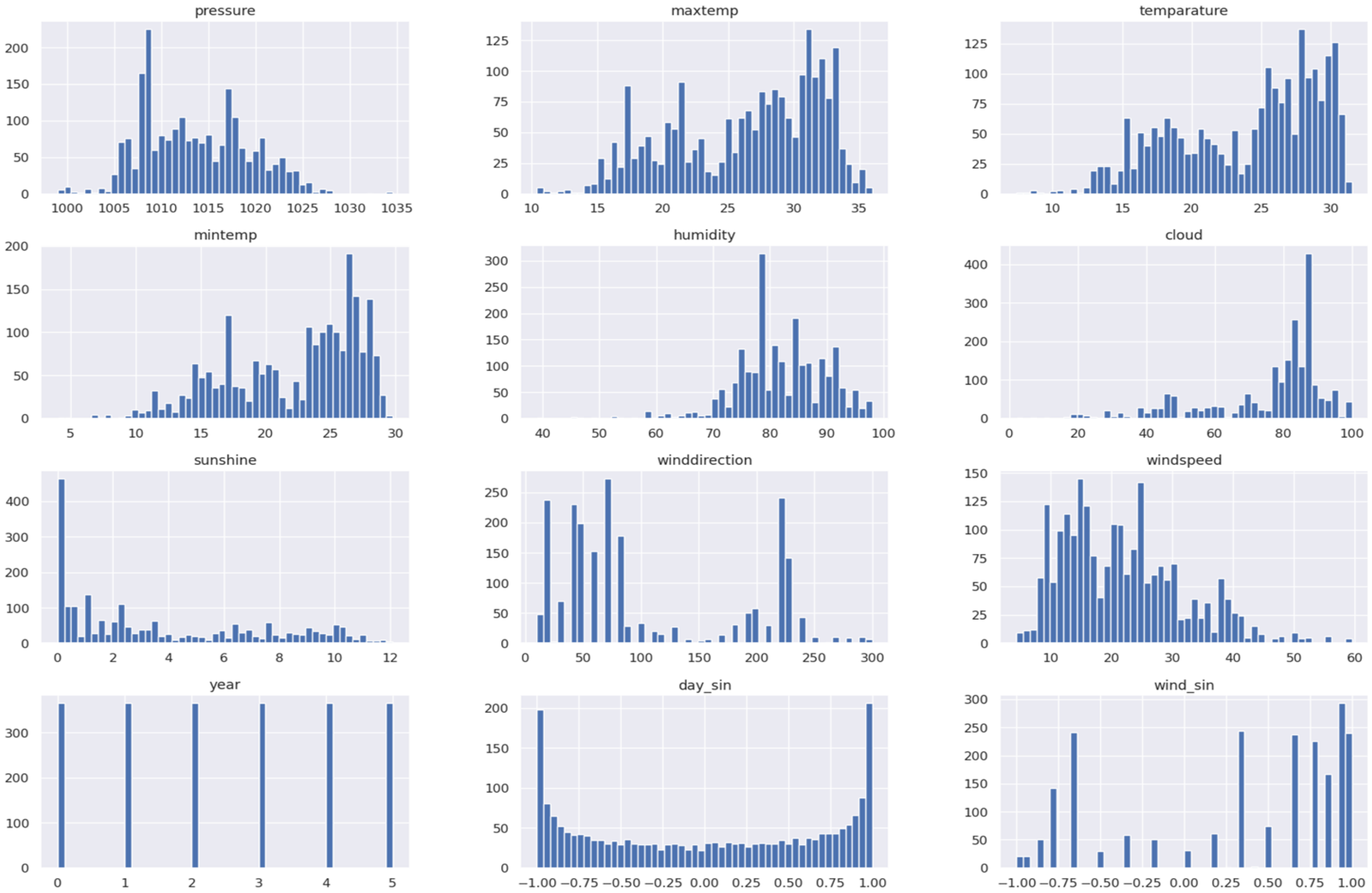
Train Data



Test Data



EDA - Distribution of Given Features (Train Data)



EDA - Distribution of Given Features (Test Data)

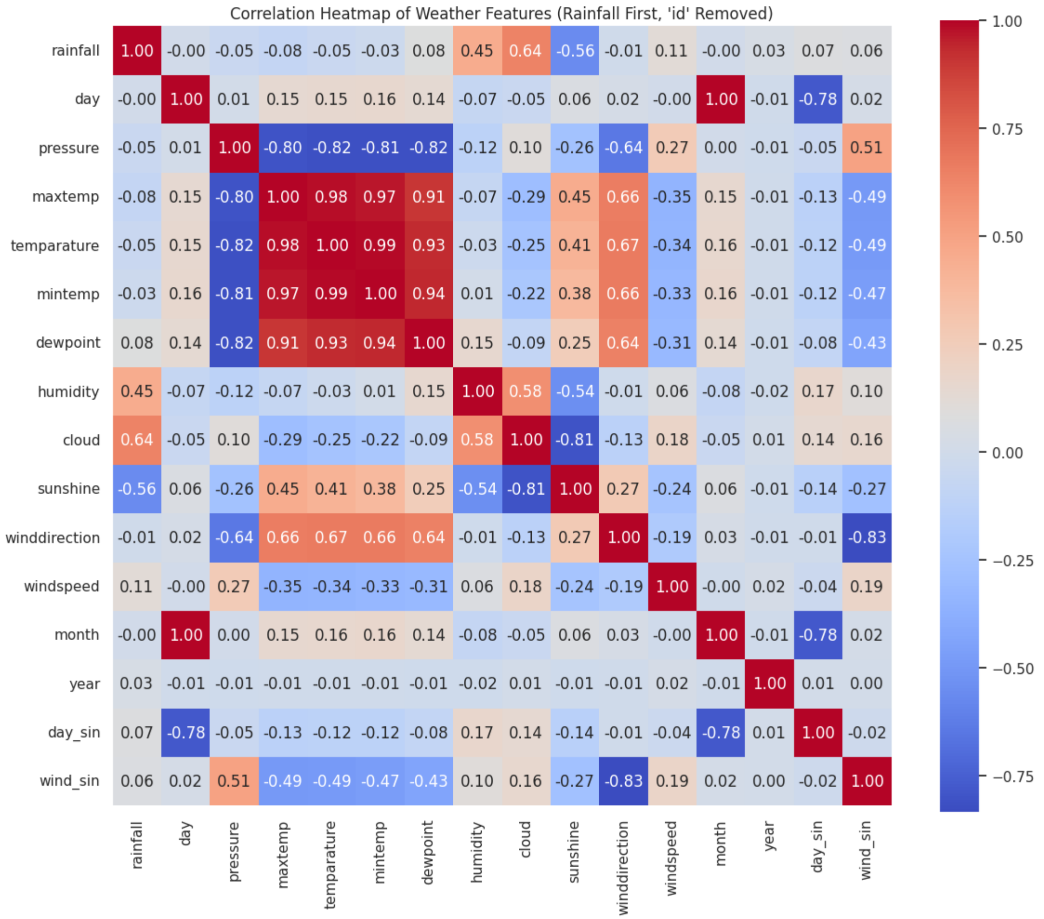
EDA - Data Preprocessing and Transformation

- Input Features: `[‘day’, 'pressure', 'maxtemp', 'temperature', 'mintemp', 'humidity', 'cloud', 'sunshine', 'winddirection', 'windspeed']`

- Additional Features:
  - wind_sin – a sine transformation of the wind direction to encode it cyclically: 2*pi*winddirection/360
  - day_sin – a sine transformation of the day to capture seasonal cycles: 2*pi*day/365
  - year – Synthetic year assigned based on index or grouping.
  - month – Extracted from the synthetic date derived from day

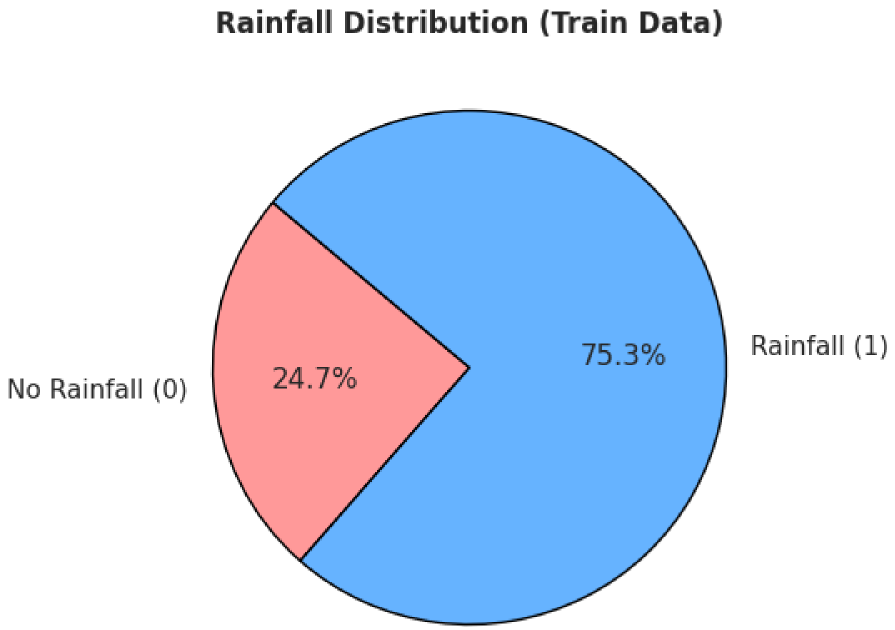
EDA - Linear Relationships

- All Features: `[‘day’, 'pressure', 'maxtemp', 'temperature', 'mintemp', 'humidity', 'cloud', 'sunshine', 'winddirection', 'windspeed', ‘month’, ‘year’, ‘wind_sin’, ‘day_sin’]`
- Relevant Features: `['pressure', 'maxtemp', 'temperature', 'mintemp', 'humidity', 'cloud', 'sunshine', 'winddirection', 'windspeed', ‘year’, ‘day_sin’, ‘wind_sin’]`
- Irrelevant Features: `[‘day’, ‘month’]`
- Additional Features to Test: `Previous day(s)’ Features`



Modeling - Baseline

- Train/Val Split
  - 80/20
  - Random / Sequential
- If a model always predicts rainfall = 1, the accuracy is 75.3%
- A basic Logistic Regression model (without feature engineering) improves the accuracy to 86%



Berkeley
UNIVERSITY OF CALIFORNIA

Modeling - Improvement

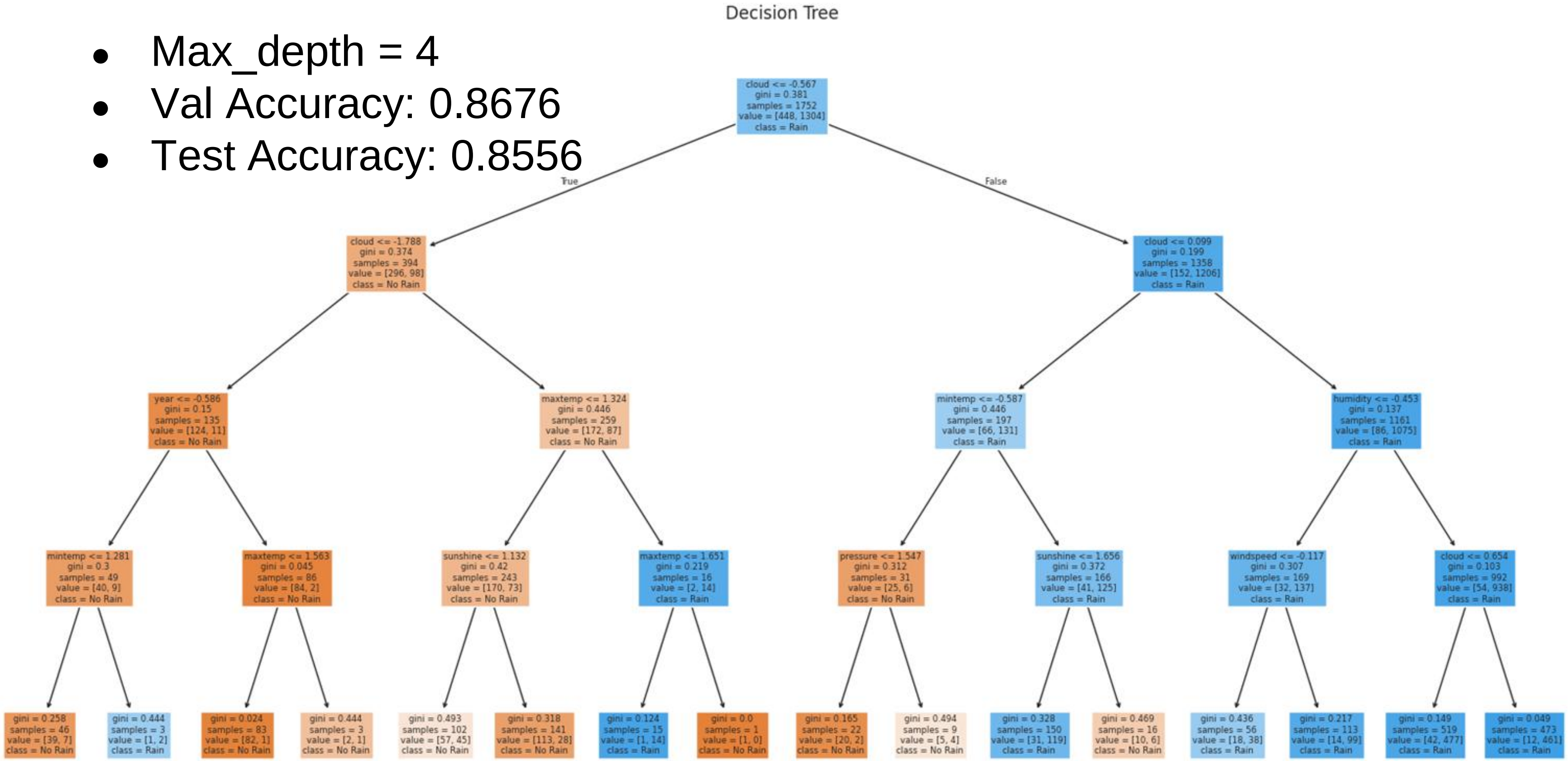
Model	Validation Accuracy	Test Score
Logistic Regression	0.8837	0.8961
KNN	0.8744	0.8716
Decision Tree	0.8676	0.8556
Random Forest	0.8744	0.8958
XGBoost	0.8721	0.8998
Neural Network	0.8790	0.9014
1D CNN	0.8833	0.8916
2D CNN	0.8787	0.8879
LSTM	0.8767	0.8959
GRU	0.8699	0.8972
Ensemble	—	**0.9038**

Experiments

- Classic Models:
  - Logistic Regression, Decision Tree, Random Forest, XGBoost, KNN
- Neural Network Models:
  - NN, 1D CNN, 2D CNN, LSTM, GRU
- Ensemble:
  - Weighted average of different models

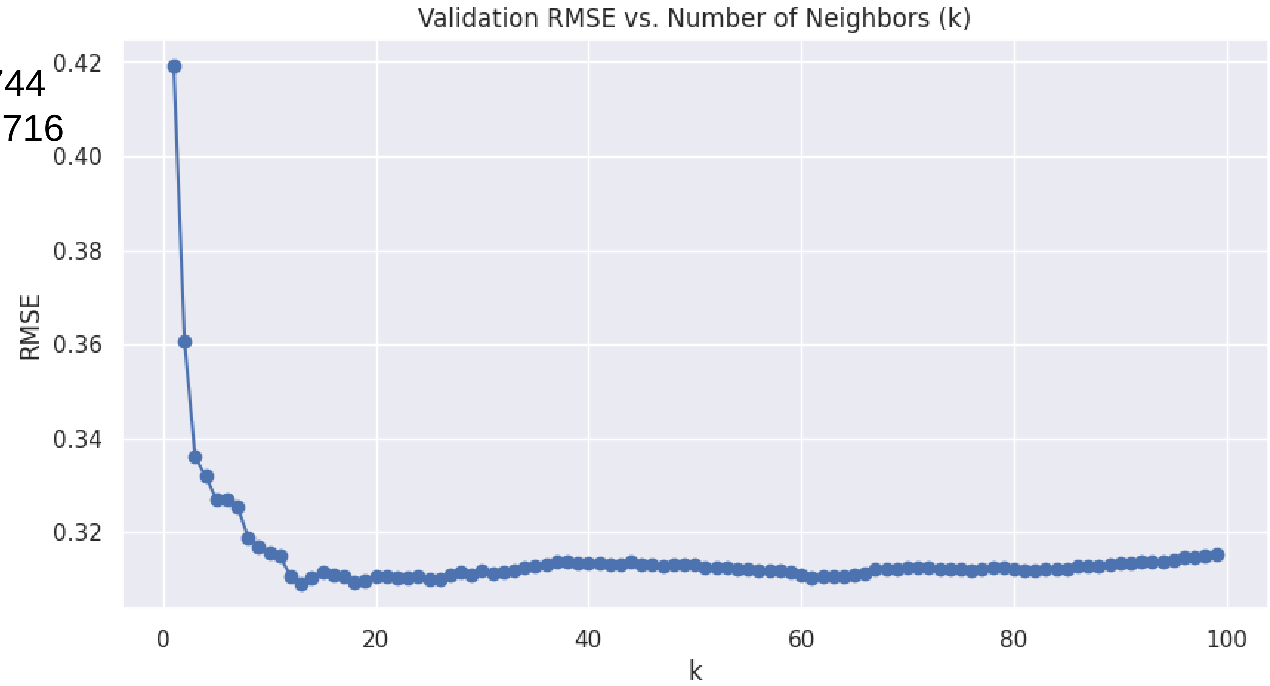
Hyperparameter Choices – Decision Trees

- Max_depth = 4
- Val Accuracy: 0.8676
- Test Accuracy: 0.8556



Hyperparameter Choices - KNN

- K = 15
- Val Accuracy: 0.8744
- Test Accuracy: 0.8716



Berkeley
UNIVERSITY OF CALIFORNIA

Hyperparameter Choices - XGBoost

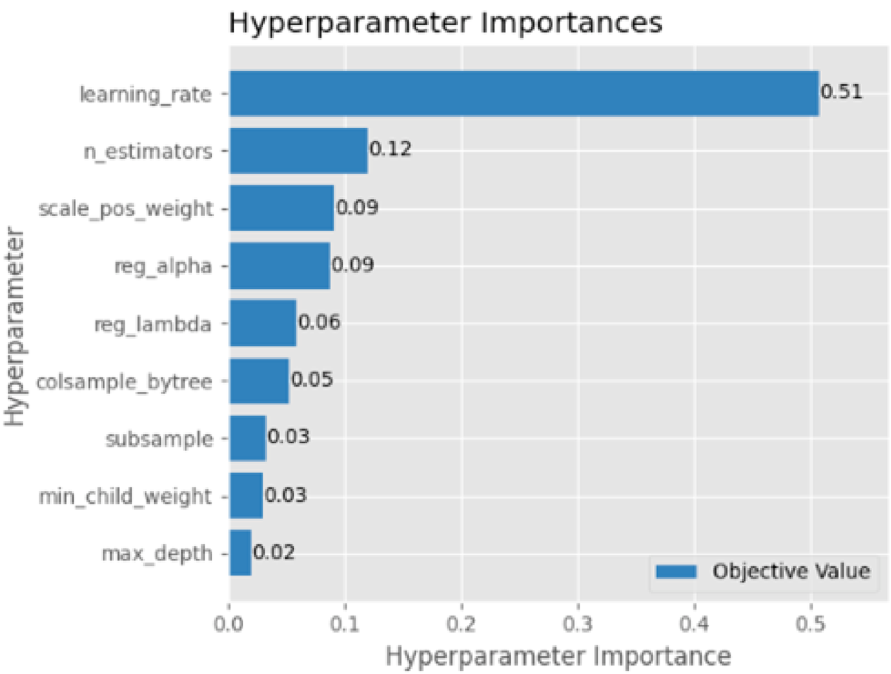
```
def objective(trial):
    params = {
        "n_estimators": trial.suggest_int("n_estimators", 100, 1000),
        "max_depth": trial.suggest_int("max_depth", 3, 10),
        "learning_rate": trial.suggest_float("learning_rate", 0.01, 0.3, log=True),
        "subsample": trial.suggest_float("subsample", 0.6, 1.0),
        "colsample_bytree": trial.suggest_float("colsample_bytree", 0.6, 1.0),
        "reg_alpha": trial.suggest_float("reg_alpha", 0.0, 1.0),
        "reg_lambda": trial.suggest_float("reg_lambda", 0.0, 10.0),
        "min_child_weight": trial.suggest_int("min_child_weight", 1, 10),
        "scale_pos_weight": trial.suggest_float("scale_pos_weight", 1.0, 10.0),
    }
```

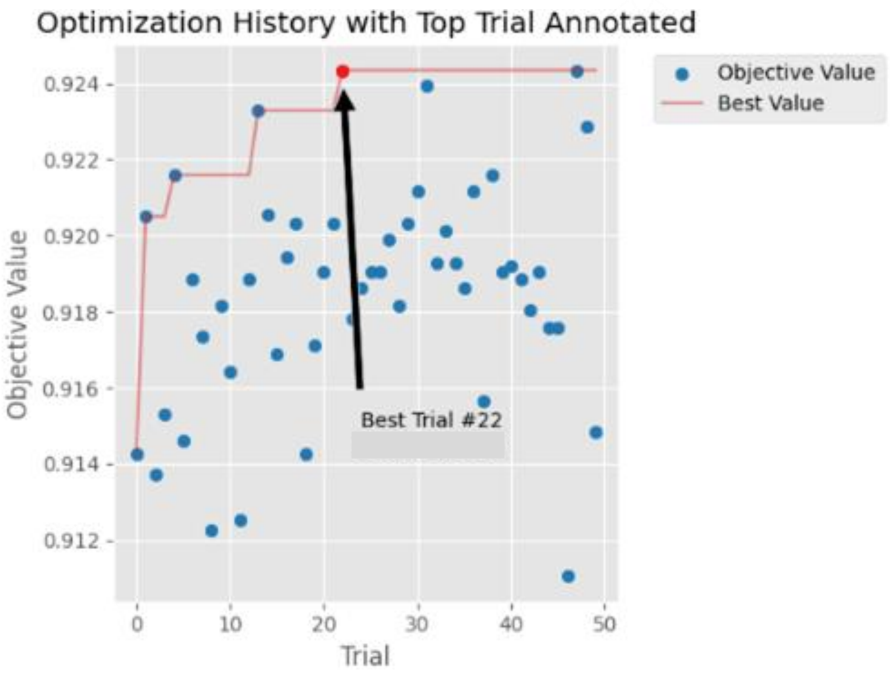
- Val Accuracy: 0.8721
- Test Accuracy: 0.8771

```
model_xgb = XGBClassifier(
    n_estimators=300,
    learning_rate=0.05,
    max_depth=6,
    subsample=0.8,
    colsample_bytree=0.8,
    scale_pos_weight=3,  # if rainfall is imbalanced
    random_state=42
)
```

Berkeley
UNIVERSITY OF CALIFORNIA

Hyperparameter Choices - XGBoost





Hyperparameter Choices - 1D CNN

```
def create_cnn_model(trial):
    model = Sequential()

    # Hyperparameters to tune
    num_filters = trial.suggest_categorical("num_filters", [32, 64, 128])
    kernel_size = trial.suggest_categorical("kernel_size", [3, 4, 5, 7])
    dropout_rate = trial.suggest_float("dropout_rate", 0.2, 0.5)
    dense_units = trial.suggest_int("dense_units", 32, 128)
    learning_rate = trial.suggest_float("learning_rate", 1e-4, 1e-2, log=True)
```

Hyperparameter Choices - 2D CNN

```
def create_2dcnn_model(trial):
    model = Sequential()

    # Trial-controlled hyperparameters
    filters = trial.suggest_categorical('filters', [32, 64, 128])
    kernel_size = trial.suggest_categorical('kernel_size', [3, 5])
    pool_size = 1
    dropout_rate = trial.suggest_float('dropout_rate', 0.2, 0.5)
    dense_units = trial.suggest_int('dense_units', 64, 256)
    learning_rate = trial.suggest_float('learning_rate', 1e-4, 1e-2, log=True)
```

Hyperparameter Choices – 1D CNN, 2D CNN

- 2D CNNs are not better than 1D CNNs
- Deeper CNNs are not better than shallower CNNs

Best 1D CNN

- Filter = 128.
- Kernel_size = 4.
- Stride = 1
- Learning_rate = 0.0005.
- Activation = ‘relu’
- Dropout rate = 0.5

Layer (type)	Output Shape	Param #
conv1d_1 (Conv1D)	(None, 8, 128)	6,272
max_pooling1d (MaxPooling1D)	(None, 4, 128)	0
dropout (Dropout)	(None, 4, 128)	0
flatten (Flatten)	(None, 512)	0
dense (Dense)	(None, 1)	513

Total params: 6,785 (26.50 KB)
Trainable params: 6,785 (26.50 KB)
Non-trainable params: 0 (0.00 B)

Hyperparameter Choices – 1D CNN

- Early_stopping
- Patience = 100
- Epochs = 300
- Batch_size 1024

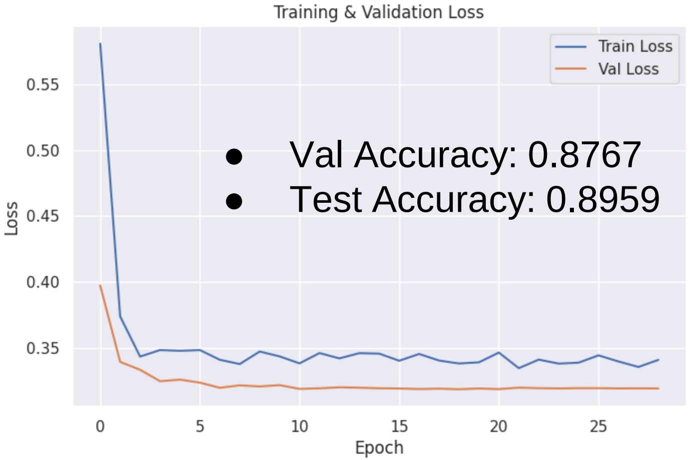
- Val Accuracy:0.8833
- Test Accuracy:0. 8916

Hyperparameter Choices - LSTM

```python
def create_lstm_model(trial):
    model = Sequential()

    units_1 = trial.suggest_int("units_1", 64, 256)
    units_2 = trial.suggest_int("units_2", 32, 128)
    dropout = trial.suggest_float("dropout", 0.2, 0.5)
    bidirectional = trial.suggest_categorical("bidirectional", [True, False])

    layer_1 = LSTM(units_1, return_sequences=True, dropout=dropout)
    if bidirectional:
        model.add(Bidirectional(layer_1))
    else:
        model.add(layer_1)

    model.add(LSTM(units_2, dropout=dropout))
    model.add(Dense(1, activation='sigmoid'))

    learning_rate = trial.suggest_float("learning_rate", 1e-4, 1e-2, log=True)
    optimizer = tf.keras.optimizers.Adam(learning_rate=learning_rate)

    model.compile(optimizer=optimizer, loss='binary_crossentropy', metrics=['accuracy'])

    return model
```

Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 1, 64)	19,712
dropout_2 (Dropout)	(None, 1, 64)	0
lstm_1 (LSTM)	(None, 64)	33,024
dropout_3 (Dropout)	(None, 64)	0
dense_2 (Dense)	(None, 128)	8,320
dropout_4 (Dropout)	(None, 128)	0
dense_3 (Dense)	(None, 1)	129

Total params: 183,557 (717.02 KB)
Trainable params: 61,185 (239.00 KB)
Non-trainable params: 0 (0.00 B)
Optimizer params: 122,372 (478.02 KB)



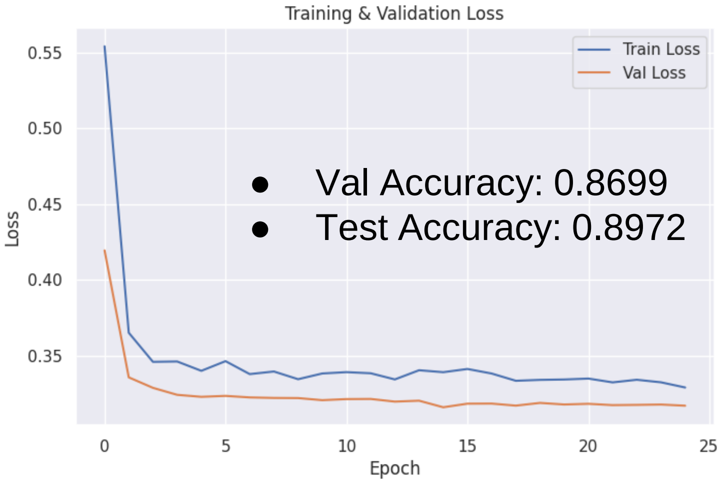
- Val Accuracy: 0.8767
- Test Accuracy: 0.8959

Hyperparameter Choices - GRU

```python
def create_model(trial):
    model = Sequential()
    units_1 = trial.suggest_int("units_1", 64, 256)
    units_2 = trial.suggest_int("units_2", 32, 128)
    dropout = trial.suggest_float("dropout", 0.2, 0.5)
    bidirectional = trial.suggest_categorical("bidirectional", [True, False])
    layer_1 = GRU(units_1, return_sequences=True, dropout=dropout)
    if bidirectional:
        model.add(Bidirectional(layer_1))
    else:
        model.add(layer_1)
    model.add(GRU(units_2, dropout=dropout, return_sequences=False))
    model.add(Dense(1, activation='sigmoid'))
    learning_rate = trial.suggest_float("learning_rate", 1e-4, 1e-2, log=True)
    optimizer = tf.keras.optimizers.Adam(learning_rate=learning_rate)
    model.compile(optimizer=optimizer, loss='binary_crossentropy', metrics=['accuracy'])
    return model
```

Layer (type)	Output Shape	Param #
gru (GRU)	(None, 1, 64)	14,976
gru_1 (GRU)	(None, 32)	9,408
dropout_5 (Dropout)	(None, 32)	0
dense_4 (Dense)	(None, 1)	33

```
Total params: 73,253 (286.15 KB)
Trainable params: 24,417 (95.38 KB)
Non-trainable params: 0 (0.00 B)
Optimizer params: 48,836 (190.77 KB)
```



- Val Accuracy: 0.8699
- Test Accuracy: 0.8972

Ensemble – Different Combinations of Models

- KNN worsened ensemble accuracy.
- LSTM and GRU also did not improve ensemble accuracy.
- A weighted average of lr, decision tree, Xgboost, NN, and CNNs results in the best score

Future Work

Incorporation of Additional Data Source

- Use satellite imagery, radar data, or real-time weather station feeds to improve prediction accuracy
- Incorporate topographical and elevation data to account for geographical rainfall patterns

Temporal and Spatial Modeling

- Extend the model to spatio-temporal predictions (e.g., where and when it will rain)
- Use recurrent neural networks (RNNs) or transformers for time-series forecasting.

Fairness

- Distribution Bias, Omitted Variable Bias, Evaluation Bias, Generalization Bias

Conclusion

- Feature engineering, particularly time-based encoding and wind direction transformation, improved all models.
- Neural models outperform simpler models on this task.
- CNNs and recurrent architectures handle sequential dependencies well.
- More complicated models are not necessarily better.
- Ensemble modeling achieved near-top leaderboard performance on Kaggle (rank 21/4382).

#	△	Team	Members	Score
1	▲ 812	Guillaume HIMBERT		0.90654
21	▲ 2242	AvasthiPrakhar		0.90376

submission_ensemble.csv **0.90376**

Complete (after deadline) · Lynne Wang · 7d ago

Berkeley
UNIVERSITY OF CALIFORNIA

References

- Kaggle : https://www.kaggle.com/competitions/playground-series-s5e3/overview
- Image: https://www.youtube.com/watch?v=Pn5NTfeKJzY
- https://www.pivotalweather.com

Contributions

- Mridul Jain: EDA, data pre-processing, lr, tree, xgboost, lstm, gru models building and tuning, slides preparation.
- Lynne Wang: EDA, data pre-processing, lr, tree, xgboost, nn, cnn, knn models building and tuning, slides preparation.
- Deepak Kumar Srivastava: EDA, data pre-processing, lr, tree, xgboost, nn, lstm models building and tuning, slides preparation.
- Naresh Kumar Chinnathambi Kailasam: EDA, data pre-processing, lr, tree, xgboost, nn, lstm models building and tuning, slides preparation.
- https://github.com/lwang9/mids-w207-final_project_team2

Berkeley
UNIVERSITY OF CALIFORNIA