Here's a **comprehensive list of key-value pairs** for `mlflow.log_param()` and `mlflow.log_params()`, categorized by machine learning tasks and frameworks. These examples cover common hyperparameters, preprocessing choices, and configurations.

---

1. General Machine Learning Parameters

Key	**Value**	**Description**
`"model_type"`	`"random_forest"`	Type of model used.
`"random_state"`	`42`	Seed for reproducibility.
`"test_size"`	`0.2`	Fraction of data for testing (e.g., `train_test_split`).
`"cross_validation"`	`True`	Whether cross-validation was used.
`"n_jobs"`	`-1`	Number of CPU cores used (`-1` = all).

---

2. Preprocessing & Feature Engineering

Key	**Value**	**Description**
`"normalize"`	`True`	Whether data was normalized.
`"scaler"`	`"StandardScaler"`	Type of scaler applied.
`"impute_strategy"`	`"median"`	Strategy for handling missing values.
`"feature_selection"`	`"PCA"`	Dimensionality reduction method.
`"n_components"`	`10`	Number of PCA components retained.

---

3. Supervised Learning (Classification/Regression)

A. Linear Models

Key	**Value**	**Description**
`"penalty"`	`"l2"`	Regularization type (e.g., Ridge/Lasso).
`"C"`	`1.0`	Inverse regularization strength.

Key	Value	Description
`"solver"`	`"lbfgs"`	Optimization algorithm.

B. Tree-Based Models

Key	Value	Description
`"n_estimators"`	`100`	Number of trees in a forest.
`"max_depth"`	`5`	Maximum depth of a tree.
`"min_samples_split"`	`2`	Minimum samples to split a node.

C. Neural Networks

Key	Value	Description
`"batch_size"`	`32`	Training batch size.
`"epochs"`	`50`	Number of training epochs.
`"optimizer"`	`"adam"`	Optimization algorithm.
`"learning_rate"`	`0.001`	Step size for weight updates.

---

4. Unsupervised Learning (Clustering/Dimensionality Reduction)

Key	Value	Description
`"n_clusters"`	`3`	Number of clusters (e.g., K-Means).
`"linkage"`	`"ward"`	Linkage method for hierarchical clustering.
`"n_neighbors"`	`15`	Neighbors for UMAP/t-SNE.

---

5. Framework-Specific Examples

A. Scikit-Learn

python Copy

```python
params = {
    "model": "RandomForestClassifier",
    "max_features": "sqrt",
    "bootstrap": True,
    "criterion": "gini"
}
mlflow.log_params(params)
```

B. TensorFlow/Keras

python

```python
mlflow.log_params({
    "loss": "categorical_crossentropy",
    "activation": "relu",
    "dropout_rate": 0.3,
    "batch_norm": True
})
```

C. PyTorch

python

```python
mlflow.log_params({
    "lr_scheduler": "StepLR",
    "momentum": 0.9,
    "weight_decay": 0.0001
})
```

D. XGBoost/LightGBM

python

```python
mlflow.log_params({
    "booster": "gbtree",
    "subsample": 0.8,
    "colsample_bytree": 0.9,
    "early_stopping_rounds": 10
})
```

---

6. Environment & Infrastructure

Key	Value	Description
`"python_version"`	`"3.8.5"`	Python version used.
`"device"`	`"GPU"`	Hardware accelerator (CPU/GPU/TPU).
`"mlflow_version"`	`"1.30.0"`	MLflow library version.

Full Code Example

python Copy

```python
import mlflow
from sklearn.ensemble import RandomForestClassifier
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split

# Load data
data = load_iris()
X_train, X_test, y_train, y_test = train_test_split(data.data, data.target, test_size=0.2)

# Define parameters
params = {
    "n_estimators": 100,
    "max_depth": 5,
    "random_state": 42,
    "criterion": "gini",
    "test_size": 0.2
}

# Train model and log
with mlflow.start_run():
    model = RandomForestClassifier(**params).fit(X_train, y_train)
    mlflow.log_params(params)  # Log all parameters at once
    mlflow.log_metric("accuracy", model.score(X_test, y_test))
    mlflow.sklearn.log_model(model, "model")
```

Key Takeaways

1. **Use descriptive names**: Keys should clearly indicate the parameter's purpose (e.g., `"random_forest_n_estimators"` instead of just `"n_estimators"`).
2. **Log all relevant parameters**: Ensures reproducibility.
3. **Avoid logging large objects**: Use `mlflow.log_artifact()` for files or `mlflow.log_model()` for models.