ML_Model_Evaluation_&_Analysis_Cheat_Sheet

August 4, 2025

1 Model Evaluation & Analysis Cheat Sheet

1.1 General Data Understanding (Use for all models)

- Mean, Median, Mode
- Standard Deviation, Variance, Coefficient of Variation
- Skewness (data shape)
- SEM (Standard Error of Mean)
- Z-Score Normalization

2 For Regression Models

2.1 Regression Evaluation

Use these when using: - Linear Regression - Decision Tree Regressor - Random Forest Regressor - SVR, XGBoost Regressor

2.1.1 Key Steps:

- Predict using: `y = mx + c`
- Check: R^2 score (how well model fits)
- Compute:
  - MAE (Mean Absolute Error)
  - MSE (Mean Squared Error)
  - RMSE (Root MSE)
  - SSR / SSE / SST
- Bias vs Variance:
  - `.score()` on train and test
- Plot Regression Line (matplotlib)

3 For Classification Models

3.1 Classification Evaluation

Use these when using: - Logistic Regression - Decision Tree Classifier - Random Forest Classifier - SVM, KNN, Naive Bayes, etc.

3.1.1 Key Metrics:

- Confusion Matrix

- Accuracy
- Precision
- Recall
- F1 Score
- ROC AUC Score (optional for binary)
- Bias vs Variance:
  - `.score()` on train and test

4 What to Use When (Table)

Step	Regression	Classification	Required?
Descriptive Stats			Always
Spread (std, var, CV)		Sometimes	Optional
Skew, SEM, Z-score			Recommended
R^2, SSR, SSE, SST			Required
Confusion Matrix			Required
Accuracy, Precision, F1			Required
MSE, RMSE, MAE			Required
Bias vs Variance			Always

Tip

- For small datasets, always inspect visually
- Use `.corr()` to detect multicollinearity
- High bias = underfitting, High variance = overfitting
- Use learning curves to visually see bias vs variance

5 REGRESSION: Predicting Salary from Experience

[2]:
```
import pandas as pd

df_reg = pd.DataFrame({
    'YearsExperience': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
    'Salary': [35000, 40000, 45000, 50000, 60000, 65000, 70000, 75000, 80000, 85000]
})
```

5.1 Descriptive Statistics

[3]:
```
print(df_reg.describe())
```

```
       YearsExperience        Salary
count         10.00000     10.000000
mean           5.50000  60500.000000
std            3.02765  17392.527131
```

```
min           1.00000  35000.000000
25%           3.25000  46250.000000
50%           5.50000  62500.000000
75%           7.75000  73750.000000
max          10.00000  85000.000000
```

5.2 Spread (std, var, CV)

[4]:
```
print("Standard Deviation:\n", df_reg.std())
print("Variance:\n", df_reg.var())
print("Coefficient of Variation:\n", df_reg.std() / df_reg.mean())
```

```
Standard Deviation:
 YearsExperience        3.027650
Salary             17392.527131
dtype: float64
Variance:
 YearsExperience    9.166667e+00
Salary             3.025000e+08
dtype: float64
Coefficient of Variation:
 YearsExperience    0.550482
Salary             0.287480
dtype: float64
```

5.3 Skewness, SEM, Z-score

[5]:
```
from scipy import stats

print("Skewness:\n", df_reg.skew())
print("SEM:\n", df_reg.sem())
print("Z-scores:\n", df_reg.apply(stats.zscore))
```

```
Skewness:
 YearsExperience    0.000000
Salary            -0.104538
dtype: float64
SEM:
 YearsExperience       0.957427
Salary             5500.000000
dtype: float64
Z-scores:
    YearsExperience    Salary
0        -1.566699 -1.545455
1        -1.218544 -1.242424
2        -0.870388 -0.939394
3        -0.522233 -0.636364
4        -0.174078 -0.030303
```

```
5         0.174078  0.272727
6         0.522233  0.575758
7         0.870388  0.878788
8         1.218544  1.181818
9         1.566699  1.484848
```

5.4 Train Linear Regression

[6]:
```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split

X = df_reg[['YearsExperience']]
y = df_reg['Salary']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)

model = LinearRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
```

5.5 R^2, SSR, SSE, SST

[7]:
```
import numpy as np

y_mean = np.mean(y_test)
SSR = np.sum((y_pred - y_mean)**2)
SSE = np.sum((y_test - y_pred)**2)
SST = SSR + SSE
print("R²:", model.score(X_test, y_test))
print("SSR:", SSR, "SSE:", SSE, "SST:", SST)
```

```
R²: 0.9990129867717004
SSR: 805962024.3757434 SSE: 789610.5826397156 SST: 806751634.9583831
```

5.6 MAE, MSE, RMSE

[8]:
```
from sklearn.metrics import mean_absolute_error, mean_squared_error

print("MAE:", mean_absolute_error(y_test, y_pred))
print("MSE:", mean_squared_error(y_test, y_pred))
print("RMSE:", np.sqrt(mean_squared_error(y_test, y_pred)))
```

```
MAE: 625.0
MSE: 394805.2913198578
RMSE: 628.3353334962612
```

5.7 Bias vs Variance

[9]:
```
print("Train R²:", model.score(X_train, y_train))
print("Test R²:", model.score(X_test, y_test))
```

```
Train R²: 0.9918138491729745
Test R²: 0.9990129867717004
```

5.8 Linear Regression – Result Explanation & Evaluation

Step 1: Descriptive Stats

Metric	YearsExperience	Salary
Mean	—	— (use df.mean())
Std Dev	3.03	17,393
Variance	9.17	3.02 Cr
CV	0.55	0.29

Criteria:

- Std Dev tells spread: smaller = tight data
- CV < 0.5 = consistent data
- Variance is high for salary due to unit size ( )

Step 2: Skewness

Feature	Skewness
Experience	0.00 → Normal
Salary	-0.10 → Slightly left-skewed

Criteria:

- 0 = normal (best)
- 0 = right-skew (some high outliers)
- < 0 = left-skew (some low outliers)

Step 3: Standard Error of Mean (SEM)

Feature	SEM
Salary	5,500

Criteria:

- SEM $< 5\%$ of mean → reliable
- Lower SEM → mean is stable

Step 4: Z-Scores

Used to normalize and detect outliers.

Example	Z-Score
Salary 85,000 → +1.48 std above mean	
Salary 35,000 → −1.55 std below mean	

Criteria:

- Z = 0 → at mean
- Z > 2 or < -2 → potential outlier

Step 5: Model Fit – R^2, SSR, SSE, SST

Metric	Value	Meaning
R^2	0.999	99.9% of salary explained by model
SSR	80.6 Cr	Variation explained
SSE	0.007 Cr (low)	Small error
SST	80.67 Cr	Total variation

Criteria:

- $R^2 > 0.9$ = excellent
- SSR » SSE = great fit
- SSE near 0 = low error

Step 6: Errors – MAE, MSE, RMSE

Metric	Value	Meaning
MAE	625	Avg absolute error
MSE	394,805	Avg squared error
RMSE	628	Similar to MAE

Criteria:

- Smaller values = better
- RMSE & MAE within small range → no large outliers

Step 7: Bias vs Variance

Metric	Value	Meaning
Train R^2	0.9918	High = low bias
Test R^2	0.9990	High = low variance

Criteria:

- Train   Test → balanced model
- Train » Test → overfitting
- Train & Test both low → underfitting

Final Conclusion

Checkpoint	Result
R^2 near 1	Excellent model
Very low MAE/RMSE	Accurate predictions
CV < 0.5 for salary	Stable data
Balanced train/test	No overfitting

Your regression model is performing perfectly on this dataset!

6 Tips

- Use this analysis after training any regression model
- Compare with other models like Decision Tree Regressor
- Add noise to test robustness

[ ]:
```
 Criteria Summary Table
Metric        Good Value        Problem Sign
Std Dev / CV        Low CV (< 0.5)        High spread
Skewness        Close to 0        > ±1 = skewed
SEM        < 5% of mean        > 10% = unstable
R²        > 0.9        < 0.7 = weak fit
MAE / RMSE        Low        High error
Train vs Test R²        Close values        Big gap = overfit
```

---

7 Classification: Predicting Pass/Fail

[10]:
```python
df_class = pd.DataFrame({
    'StudyHours': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
    'Attendance': [50, 55, 60, 65, 70, 75, 80, 85, 90, 95],
    'Pass': [0, 0, 0, 0, 1, 1, 1, 1, 1, 1]
})
```

7.1 Descriptive Statistics

[11]:
```python
print(df_class.describe())
```

```
       StudyHours  Attendance       Pass
count    10.00000   10.000000  10.000000
mean      5.50000   72.500000   0.600000
std       3.02765   15.138252   0.516398
```

```
min       1.00000   50.000000   0.000000
25%       3.25000   61.250000   0.000000
50%       5.50000   72.500000   1.000000
75%       7.75000   83.750000   1.000000
max      10.00000   95.000000   1.000000
```

7.2 Spread (optional)

[12]:
```python
print(df_class.std())
print(df_class.var())
```

```
StudyHours     3.027650
Attendance    15.138252
Pass           0.516398
dtype: float64
StudyHours      9.166667
Attendance    229.166667
Pass            0.266667
dtype: float64
```

7.3 Skew, SEM, Z-score

[13]:
```python
print("Skew:\n", df_class.skew())
print("SEM:\n", df_class.sem())
print("Z-score:\n", df_class.apply(stats.zscore))
```

```
Skew:
 StudyHours    0.000000
Attendance    0.000000
Pass         -0.484123
dtype: float64
SEM:
 StudyHours    0.957427
Attendance    4.787136
Pass          0.163299
dtype: float64
Z-score:
    StudyHours  Attendance      Pass
0   -1.566699   -1.566699 -1.224745
1   -1.218544   -1.218544 -1.224745
2   -0.870388   -0.870388 -1.224745
3   -0.522233   -0.522233 -1.224745
4   -0.174078   -0.174078  0.816497
5    0.174078    0.174078  0.816497
6    0.522233    0.522233  0.816497
7    0.870388    0.870388  0.816497
8    1.218544    1.218544  0.816497
9    1.566699    1.566699  0.816497
```

7.4 Train Logistic Regression

[14]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split

X = df_class[['StudyHours', 'Attendance']]
y = df_class['Pass']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
 ↪random_state=42)

clf = LogisticRegression()
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
```

7.5 Confusion Matrix

[15]:
```
from sklearn.metrics import confusion_matrix

print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
```

```
Confusion Matrix:
 [[1 0]
 [0 2]]
```

7.6 Accuracy, Precision, Recall, F1

[16]:
```
from sklearn.metrics import accuracy_score, precision_score, recall_score,
 ↪f1_score

print("Accuracy:", accuracy_score(y_test, y_pred))
print("Precision:", precision_score(y_test, y_pred))
print("Recall:", recall_score(y_test, y_pred))
print("F1 Score:", f1_score(y_test, y_pred))
```

```
Accuracy: 1.0
Precision: 1.0
Recall: 1.0
F1 Score: 1.0
```

7.7 Bias vs Variance

[17]:
```
print("Train Score:", clf.score(X_train, y_train))
print("Test Score:", clf.score(X_test, y_test))
```

```
Train Score: 1.0
Test Score: 1.0
```

7.8 Logistic Regression – Result Explanation & Evaluation

1. Descriptive Statistics

Feature	Mean	Std Dev	Min	25%	50%	75%	Max
StudyHours	5.5	3.03	1.0	3.25	5.5	7.75	10.0
Attendance	72.5	15.14	50	61.25	72.5	83.75	95
Pass	0.6	0.52	0	0	1	1	1

Criteria:

- Balanced values, no extreme outliers
- Binary target variable (0 = Fail, 1 = Pass)

2. Spread Measures (Optional)

Feature	Variance	Coefficient of Variation (CV)
StudyHours	9.17	0.55 (Moderate spread)
Attendance	229.17	0.21 (Low spread = stable)
Pass	0.27	0.86 (Binary target variable)

CV $< 0.5 \rightarrow$ stable
CV $> 0.5 \rightarrow$ some variability

3. Skewness, SEM, Z-score

Skewness

Feature	Skewness	Interpretation
StudyHours	0.00	Normal distribution
Attendance	0.00	Normal distribution
Pass	-0.48	Slight left skew

Skew between -1 and +1 → Acceptable

SEM (Standard Error of Mean)

Feature	SEM	Interpretation
StudyHours	0.96	Low → mean is stable
Attendance	4.79	Stable mean
Pass	0.16	Target class is moderately stable

SEM < 5–10% of mean is reliable

Z-score

Example: | StudyHours = 10 → Z = +1.57
| Pass = 0 → Z = -1.22
| Pass = 1 → Z = +0.82

Helps detect outliers and normalize data.

4. Model Performance – Confusion Matrix

```
[[1 0]
 [0 2]]
```

	Predicted 0	Predicted 1
Actual 0 (Fail)	1	0
Actual 1 (Pass)	0	2

100% correct predictions → perfect model (on this test set)

** 5. Metrics – Accuracy, Precision, Recall, F1**

Metric	Value	Meaning
Accuracy	1.0	All predictions correct
Precision	1.0	No false positives
Recall	1.0	No false negatives
F1 Score	1.0	Perfect balance

Criteria for Classification Metrics

Metric	Good Value	Issue If
Accuracy	> 0.9	Low with class imbalance
Precision	> 0.8	False positives increase
Recall	> 0.8	False negatives increase
F1 Score	~ Precision & Recall	Too low = imbalance

6. Bias vs Variance

Metric	Score	Meaning
Train Score	1.0	Model fits training data perfectly
Test Score	1.0	Generalizes perfectly (for now)

Train Test = **No overfitting**

Final Conclusion

Observation	Verdict
Descriptive stats balanced	Good distribution

Observation	Verdict
SEM and Skew in control	Reliable stats
Model Metrics = All 1.0	Perfect classification
Bias = Variance	No overfitting

Tip: Add more **realistic or confusing samples** to test robustness.

Recommendation

Use this as your checklist:

- Descriptive stats
- Skew, SEM, Z-score
- Confusion matrix
- Accuracy, precision, recall, F1
- Train vs test score

Then decide if your model is:

- Underfitting
- Overfitting
- Or well generalized