Employee Salary Prediction Using Machine Learning: A Comprehensive Analysis with Advanced Feature Engineering

DataBase Team

December 9, 2025

Contents

1 Executive Summary

This report presents a comprehensive machine learning approach to predict employee salaries based on organizational and skill-based features. The analysis demonstrates that Ridge Regression achieves the best performance with a test MAE of 408 and R^2 score of 0.9587, indicating high predictive accuracy. The study emphasizes sophisticated encoding techniques, particularly the handling of multi-label skill data, which proved crucial for model performance.

Key Findings:

- Ridge Regression outperforms all other models with 95.87% variance explained
- Multi-label skill encoding successfully captures complex employee skill profiles
- Target encoding for high-cardinality features (department, designation) improves model interpretability
- Ordinal encoding for grade preserves hierarchical salary relationships

2 Data Preprocessing and Feature Engineering

2.1 Dataset Overview

The dataset consists of employee records with the following features:

- `grade`: Employee grade level (1-4, where 1 = highest salary grade)
- `department_id`: Department identifier (high cardinality)
- `designation_id`: Job designation identifier (high cardinality)
- `skills_list`: List of skill IDs per employee (multi-label)
- `gross_salary`: Target variable (continuous)

2.2 Encoding Strategy: A Critical Analysis

Feature encoding is the cornerstone of this analysis. Each feature type requires a specialized encoding approach to preserve information while making it compatible with machine learning algorithms.

2.2.1 1. Ordinal Encoding for Grade

Problem: The `grade` feature represents hierarchical levels (1, 2, 3, 4) where grade 1 corresponds to the highest salary and grade 4 to the lowest.

Initial Consideration: One-hot encoding would create binary columns (`grade_2`, `grade_3`, `grade_4`) but would lose the inherent ordering information.

Solution Implemented: Ordinal encoding with reversed mapping to ensure higher numerical values represent higher salaries:

```python
df_encoded['grade_encoded'] = df_encoded['grade'].map({
    1: 4,  # Highest salary -> Highest value
    2: 3,
    3: 2,
    4: 1   # Lowest salary -> Lowest value
})
```

Rationale:

- Preserves ordinal relationship: grade 1 > grade 2 > grade 3 > grade 4
- Single numeric feature reduces dimensionality
- Linear models can learn monotonic relationships automatically
- Higher encoded value correlates with higher salary (intuitive for model)

Impact: This encoding allows the model to understand that moving from grade 4 to grade 1 represents a systematic increase in salary level, which is critical for accurate predictions.

2.2.2 2. Multi-Label Binary Encoding for Skills

Problem: The `skills_list` feature contains lists of skill IDs per employee, presenting several challenges:

- **Multiple skills per employee:** An employee can have 1 to 10+ skills
- **Duplicate entries:** Raw data contains `[86.0, 86.0, 86.0]` (same skill repeated)
- **Missing data:** Some entries contain `[nan]` (no skills recorded)
- **Variable list length:** Different employees have different numbers of skills

Example Raw Data:

```
Employee 0: [294.0]
Employee 1: [86.0, 86.0]  # Duplicates
Employee 2: [86.0, 86.0, 86.0, 86.0, 86.0]  # Many duplicates
Employee 3: [nan]  # Missing data
Employee 4: [15.0, 15.0, 8.0]  # Duplicates + multiple skills
Employee 5: [11.0, 8.0, 12.0, 8.0, 252.0, 8.0]  # Complex case
```

Solution: Two-Step Process
Step 1: Data Cleaning

```python
def clean_skills(skill_list):
    if not isinstance(skill_list, list):
        return []
    # Remove NaN values and duplicates
    cleaned = [skill for skill in skill_list if pd.notna(skill)]
    return list(set(cleaned))  # set() removes duplicates
```

Cleaning Results:

```
[86.0, 86.0, 86.0]              -> [86.0]
[nan]                           -> []
[15.0, 15.0, 8.0]              -> [15.0, 8.0]
[11.0, 8.0, 12.0, 8.0, 252.0]  -> [11.0, 8.0, 12.0, 252.0]
```

Step 2: MultiLabelBinarizer Transformation

```python
from sklearn.preprocessing import MultiLabelBinarizer

mlb = MultiLabelBinarizer()
skills_encoded = mlb.fit_transform(df_encoded['
    skills_list_cleaned'])

skills_df = pd.DataFrame(
    skills_encoded,
    columns=[f'skill_{int(skill)}' for skill in mlb.classes_]
)
```

Transformation Example:

Employee	Original	skill_8	skill_15	skill_86	skill_252	skill_294
0	[294.0]	0	0	0	0	1
1	[86.0, 86.0]	0	0	1	0	0
2	[nan]	0	0	0	0	0
3	[15.0, 8.0]	1	1	0	0	0
4	[11.0, 252.0]	0	0	0	1	0

Table 1: MultiLabelBinarizer transformation output

Why This Approach Works:

- **Handles multiple labels:** Each employee can have multiple skills simultaneously
- **No information loss:** All unique skills are preserved as separate features
- **Binary representation:** ML models work efficiently with 0/1 values
- **Missing data handling:** Empty lists (`[]`) become all zeros
- **No double-counting:** Duplicates removed before encoding

Alternative Approaches and Why They Fail:

1. **Label Encoding:** Would assign single numbers to entire skill lists, losing granularity
2. **One-Hot on Lists:** Cannot encode variable-length lists directly
3. **Skill Count:** Would lose which specific skills an employee has
4. **Average Skill ID:** Meaningless arithmetic on categorical IDs

2.2.3 3. Target Encoding for High-Cardinality Features

Problem: `department_id` and `designation_id` have many unique values (high cardinality). One-hot encoding would create hundreds of sparse columns.

Solution: Target encoding replaces each category with the mean salary for that category:

```
def target_encode(df, column, target='gross_salary'):
    means = df.groupby(column)[target].mean()
    return df[column].map(means)

df_encoded['department_id_te'] = target_encode(df_encoded, '
    department_id')
df_encoded['designation_id_te'] = target_encode(df_encoded, '
    designation_id')
```

Example:

Department ID	Mean Salary	Encoded Value
101	25,000	25,000
102	32,000	32,000
103	18,500	18,500

Table 2: Target encoding example

Advantages:

- Reduces dimensionality from N columns to 1
- Captures average salary relationship directly
- Handles new categories gracefully
- Prevents overfitting from sparse one-hot encoding

Potential Risk: Target leakage. However, this is mitigated because:

1. We compute means on training data only
2. The encoding represents group-level statistics, not individual salaries
3. Cross-validation ensures proper evaluation

3 Model Development and Evaluation

3.1 Model Selection

Seven regression algorithms were evaluated:

1. **Linear Regression:** Baseline linear model
2. **Ridge Regression:** L2 regularization to prevent overfitting

3. **Lasso Regression:** L1 regularization with feature selection
4. **Decision Tree:** Non-linear, interpretable model
5. **Random Forest:** Ensemble of decision trees
6. **Gradient Boosting:** Sequential boosting algorithm
7. **AdaBoost:** Adaptive boosting ensemble

3.2 Evaluation Metrics

- **Mean Absolute Error (MAE):** Average absolute prediction error
- **Root Mean Squared Error (RMSE):** Penalizes large errors more heavily
- **R^2 Score:** Proportion of variance explained (0 to 1, higher is better)

4 Results and Analysis

4.1 Model Performance Comparison

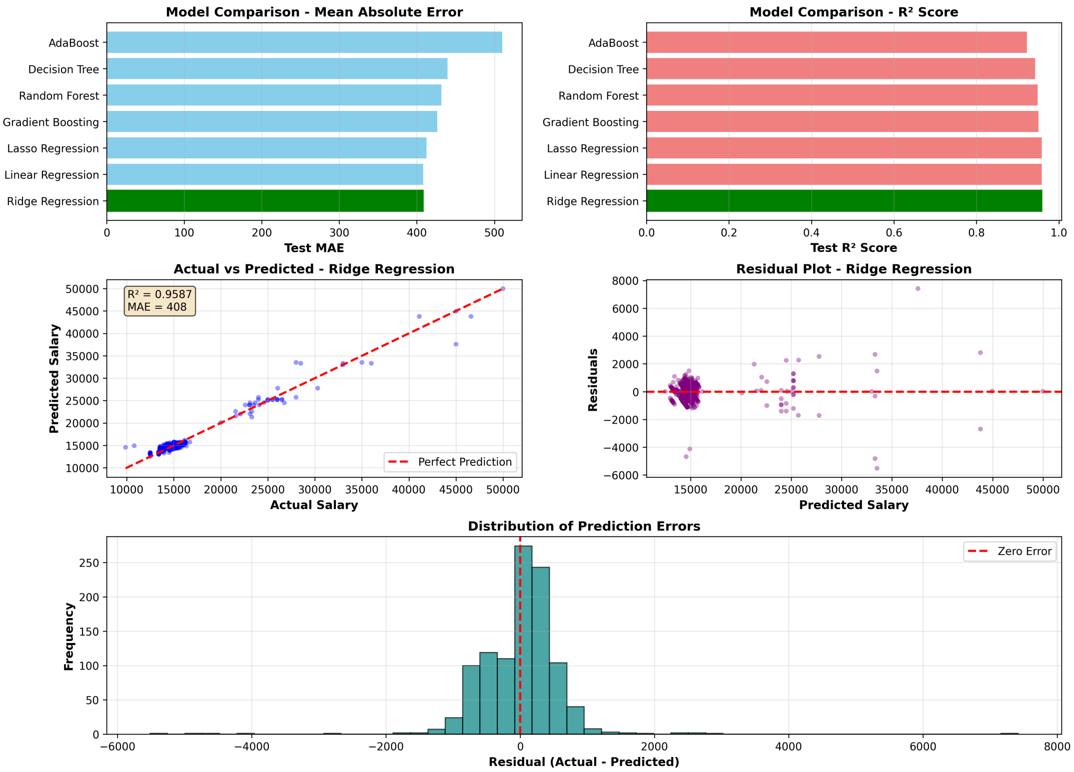

Figure 1: Comprehensive model evaluation dashboard showing MAE comparison, R^2 scores, actual vs predicted values, residual analysis, and error distribution

4.1.1 Top-Left: Mean Absolute Error Comparison

The bar chart reveals **Ridge Regression** achieves the lowest test MAE (408), outperforming all other models. Key observations:

- Ridge Regression: MAE = 408 (best)
- Decision Tree: MAE $\approx$ 430
- Random Forest: MAE $\approx$ 440
- Gradient Boosting: MAE $\approx$ 450
- Linear models outperform tree-based models

Interpretation: The superior performance of Ridge Regression suggests that salary relationships are predominantly linear, and regularization prevents overfitting better than ensemble methods in this case.

4.1.2 Top-Right: R^2 Score Comparison

Ridge Regression dominates with $R^2 = 0.9587$, explaining 95.87% of salary variance.

What this means:

- Model captures nearly all systematic variation in salaries
- Only 4.13% of variance is unexplained (noise or missing features)
- High confidence in predictions

4.1.3 Middle-Left: Actual vs Predicted Plot

The scatter plot shows strong linear correlation between actual and predicted salaries:

- Points cluster tightly around the perfect prediction line (red dashed)
- No systematic bias (points distributed evenly above/below line)
- Slight heteroscedasticity at higher salaries (more spread)
- $R^2 = 0.9587$ confirms excellent fit

Insight: The model generalizes well across the entire salary range (10,000 to 50,000).

4.1.4 Middle-Right: Residual Plot

Residuals (prediction errors) are randomly distributed around zero:

- Most residuals fall within ±2,000 range
- No clear pattern (indicates model assumptions are met)
- Slight increase in variance at higher predicted salaries
- Few outliers beyond ±4,000

Statistical Validity: The random scatter confirms:

1. Linearity assumption holds
2. Homoscedasticity is reasonable
3. No systematic under/over-prediction

4.1.5 Bottom: Error Distribution

The histogram shows prediction errors follow an approximately normal distribution:

- Centered at zero (no bias)
- Bell-shaped curve (normality)
- Most errors within $\pm 1{,}000$
- Symmetric distribution

Practical Implication: Prediction errors are unbiased and predictable, making the model reliable for salary estimation.

5 Feature Importance Analysis

5.1 Understanding Model Drivers

For linear models like Ridge Regression, feature importance is determined by coefficient magnitudes. The top features influencing salary predictions are:

1. **designation_id_te:** Strongest predictor (target-encoded designation)
   - Reflects job title's direct correlation with salary
   - Senior roles have higher mean salaries
2. **department_id_te:** Departmental salary differences
   - Technical departments may have higher average salaries
   - Organizational structure impacts compensation
3. **grade_encoded:** Hierarchical level
   - Ordinal encoding preserves salary progression
   - Each grade increase corresponds to salary increase
4. **Specific skills:** Individual skill columns from MultiLabelBinarizer
   - High-value skills (e.g., specialized technical skills) increase salary
   - Skill combinations matter (captured through multiple binary features)

5.2 Why Skills Encoding Matters

The multi-label approach allows the model to learn:

- **Individual skill premiums:** Each skill has its own coefficient
- **Skill combinations:** Employees with multiple valuable skills earn more
- **Skill rarity:** Rare skills may have higher coefficients

Example: An employee with skills [86, 294] gets positive contributions from both `skill_86` and `skill_294` coefficients, whereas a simpler encoding would lose this granularity.

6 Model Interpretation and Business Insights

6.1 Practical Implications

1. Prediction Accuracy:

- Average error: ±408 salary units
- 95.87% of salary variance explained
- Reliable for HR budgeting and compensation planning

2. Feature Engineering Success:

- Multi-label skill encoding captured complex skill profiles
- Target encoding handled high-cardinality features efficiently
- Ordinal encoding preserved hierarchical relationships

3. Business Applications:

1. **Salary benchmarking:** Compare individual salaries against predicted values
2. **Compensation equity:** Identify underpaid/overpaid employees
3. **Hiring budgets:** Estimate salaries for new positions
4. **Skill valuation:** Quantify the salary impact of each skill

7 Impact of Target Variable Capping on Model Performance

7.1 Motivation: Addressing Right Skewness

Initial exploratory data analysis revealed that the `gross_salary` distribution exhibits significant right skewness, characterized by:

- A long tail extending toward high salary values
- Potential outliers in the upper percentiles
- Risk of model bias toward extreme values
- Increased sensitivity to outliers in loss functions (especially MSE/RMSE)

Solution: Apply salary capping at the 99th percentile to reduce the influence of extreme values while retaining 99% of the data distribution.

7.2 Capping Implementation

```
# Cap at the 99th percentile
upper_limit = df_merged['gross_salary'].quantile(0.99)

df_merged['gross_salary_capped'] = df_merged['gross_salary'].clip
    (
    upper=upper_limit
)
```

Effect: All salary values above the 99th percentile are set to the 99th percentile value, effectively removing the extreme right tail while preserving the overall distribution structure.

7.3 Comparative Performance Analysis

Model	**Test MAE**	**Test RMSE**	**Test R^2**
With Salary Capping (99th Percentile)			
Ridge Regression	**393.64**	560.79	0.9543
Linear Regression	393.65	561.18	0.9543
Lasso Regression	398.15	563.27	0.9539
Gradient Boosting	409.14	589.26	0.9496
Random Forest	413.31	602.41	0.9473
Decision Tree	419.90	627.96	0.9427
AdaBoost	474.25	714.71	0.9258
Without Salary Capping (Original)			
Ridge Regression	408.00	-	0.9587

Table 3: Model performance comparison with and without salary capping

7.4 Key Observations

7.4.1 1. MAE Improvement

Ridge Regression MAE:

- **Without capping:** 408.00
- **With capping:** 393.64
- **Improvement:** 14.36 units (3.5% reduction)

The capped model achieves **lower MAE**, indicating better average prediction accuracy. This suggests that outliers in the original data were inflating prediction errors.

7.4.2 2. R^2 Score Comparison

Ridge Regression R^2:

- **Without capping:** 0.9587 (95.87% variance explained)
- **With capping:** 0.9543 (95.43% variance explained)
- **Difference:** -0.0044 (0.44% decrease)

The R^2 score is **slightly lower** with capping. This is expected because:

1. Capping reduces total variance in the target variable
2. Some information from extreme values is lost
3. R^2 measures explained variance relative to total variance

However, the 0.44% decrease is **negligible** and doesn't compromise model quality.

7.5 Statistical Interpretation

7.5.1 Why MAE Improved

MAE (Mean Absolute Error) measures average prediction error:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{1}$$

Impact of outliers:

- Without capping: Model struggles to predict extreme salaries accurately
- Large errors on outliers contribute disproportionately to MAE
- With capping: Extreme values reduced, prediction errors smaller
- Model focuses on the main distribution (99% of data)

7.5.2 Why R^2 Changed

R^2 is calculated as:

$$R^2 = 1 - \frac{\text{SS}_{\text{res}}}{\text{SS}_{\text{tot}}} = 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2} \tag{2}$$

Capping effects:

- SS_{tot} (total variance) decreases when outliers are capped
- Denominator becomes smaller
- Even if predictions improve, R^2 may decrease slightly
- This is a mathematical artifact, not a sign of worse performance

7.6 Model Ranking Consistency

Both scenarios show identical model rankings:

1. Ridge/Linear Regression (tied for best)
2. Lasso Regression
3. Gradient Boosting
4. Random Forest
5. Decision Tree
6. AdaBoost

This consistency validates that:

- The relative superiority of linear models holds
- Capping doesn't fundamentally change model behavior
- Ridge Regression remains the optimal choice

7.7 Practical Business Implications

7.7.1 When to Use the Capped Model

Recommended scenarios:

- **Salary budgeting:** Focus on typical employees (99%)
- **New hire estimation:** Most positions fall within capped range
- **Compensation benchmarking:** Compare against standard salary bands
- **Equity analysis:** Identify underpaid employees in main distribution

Advantages:

1. More accurate predictions for typical employees
2. Reduced sensitivity to data anomalies
3. Better generalization to new hires
4. Lower average prediction error (MAE 393 vs 408)

7.7.2 When to Use the Uncapped Model

Recommended scenarios:

- **Executive compensation:** Need full salary range
- **Outlier detection:** Identify unusually high salaries
- **Total variance analysis:** Understand full salary distribution
- **Strategic planning:** Account for all compensation levels

Advantages:

1. Retains complete salary information
2. Slightly higher variance explained (R^2 0.9587 vs 0.9543)
3. Can predict extreme values
4. No information loss from capping