> 💡 **Recommended Environment**:
> Run this notebook in the `model_eval_suite` Conda environment for best results.
> See setup instructions in the Usage Guide.
>
> ⚠️ If you're running this outside Conda, you can install dependencies manually: Uncomment the line below to install from the root requirements file.

```
# !pip install -r ../../requirements.txt
```

🧪 Model Evaluation Suite Demo Notebook

This notebook demonstrates how to use the **Model Evaluation Suite** to:

- Prepare and validate input data.
- Run a full modeling pipeline using YAML configuration files.
- Log models and artifacts with MLflow.
- Evaluate a production candidate against a holdout dataset.
- Optionally compare against a **baseline model** for performance drift or uplift.

▶ 📦 Project Structure
▶ ⚙️ Workflow Overview

📜 Configuration Setup

This notebook is driven by modular **YAML configuration files**, which serve as the central control system for the evaluation suite.

▶ click here to expand section

🔧 Custom Feature Engineering

This suite supports plug-and-play custom transformers via the `feature_engineering` block in your YAML.

▶ click here to expand

📚 Dashboard Guidence

📉 Pre-Model Diagnostics Dashboard

This optional module runs before any model training or validation occurs. It provides key insights into the integrity and statistical structure of your input data. It is driven by the `pre_model_diagnostics` block in your YAML and is best used in notebook workflows.

▶ click here to expand section

📊 Model Evaluation Dashboard

This dashboard provides an interactive summary of the model trained in the experimental run. It visualizes performance on the test set and includes explainability tools to support model diagnostics and stakeholder communication.

▶ Summary
▶ Details
▶ Importance
▶ Explainability
▶ Plotviewers
▶ Metadata
▶ Alerts

📌 Core Imports

You can access the main runners directly from the package thanks to a clean interface exposed via `__init__.py`. These entrypoints allow you to run each stage of the pipeline from a single import.

In [1]:
```
from model_eval_suite import run_experiment, validate_champion, prep_data
```

In [2]:
```
import os
# Change the working directory to the project root
os.chdir('..')
```

🔽 Data Preparation

If you're starting from raw CSVs, you can use the suite's built-in preprocessing tool, `data_prep.py` to split the data into training, testing, and holdout sets.

Skip this if you've already created your `train.csv`, `test.csv`, and `holdout.csv`.

In [3]:
```
prep_data(config_path="config/data_prep.yaml")
```

```
Loading raw data from: data/input_data/salifort_50k.csv
Performing initial holdout split...
Performing train/test split on development data...
```

```
✅ Train data saved to: data/dev_data/train_data.csv (30000 rows)
✅ Test data saved to: data/dev_data/test_data.csv (10000 rows)
✅ Holdout data saved to: data/holdout_data/holdout_data.csv (10000 rows)
```

⚙️ Model Experiment Runs (Demo)

This demo walks through multiple model runs using the `salifort_50k` dataset. Although the dataset is optimized for **classification tasks**, it is used for both classification and regression pipelines to demonstrate flexibility and YAML-driven control.

We run the following models using the evaluation suite:

🔍 Classifier Models

- **Gaussian Naive Bayes**
  Config: config/classifier/guas_nb.yaml
- **Logistic Regression**
  Config: config/classifier/logreg.yaml
- **XGBoost Classifier**
  Config: config/classifier/xgboost.yaml

📈 Regressor Models

- **Linear Regression**
  Config: config/regressor/linreg.yaml
- **XGBoost Regressor**
  Config: config/regressor/xgboost_reg.yaml

Each model triggers:

- An optional **Pre-Model Diagnostics Dashboard** (if enabled in YAML)
- A complete **Evaluation Dashboard** with explainability, distributions, and exportable artifacts

At the end of the demo, we use the **champion validation** system to validate and crown the two XGBoost models — one for classification, one for regression.

> All behavior is controlled by the YAML configs. See the `config/` directory or `config_resources/config.zip` for template downloads.

In [4]:
```
# ========== Naive - Bayes ==========
run_experiment(user_config_path="config/classifier/guas_nb.yaml")
```

```
Registered model 'gnb_demo_01' already exists. Creating a new version of thi
s model...
Created version '4' of model 'gnb_demo_01'.
⚠️ SHAP explainer creation failed: The passed model is not callable and cann
ot be analyzed directly with the given masker! Model: GaussianNB()
✅ Run complete: `gnb_demo_01`
📁 Artifacts saved to:
   - Plots:   exports/plots/gnb_demo_01
   - Reports: exports/reports/gnb_demo_01
📦 MLflow model: `gnb_demo_01`

--- Rendering Dashboards ---
```

🔍 Pre-Modeling Diagnostics

Overview | Missingness | Collinearity | Distributions | Eval Plots

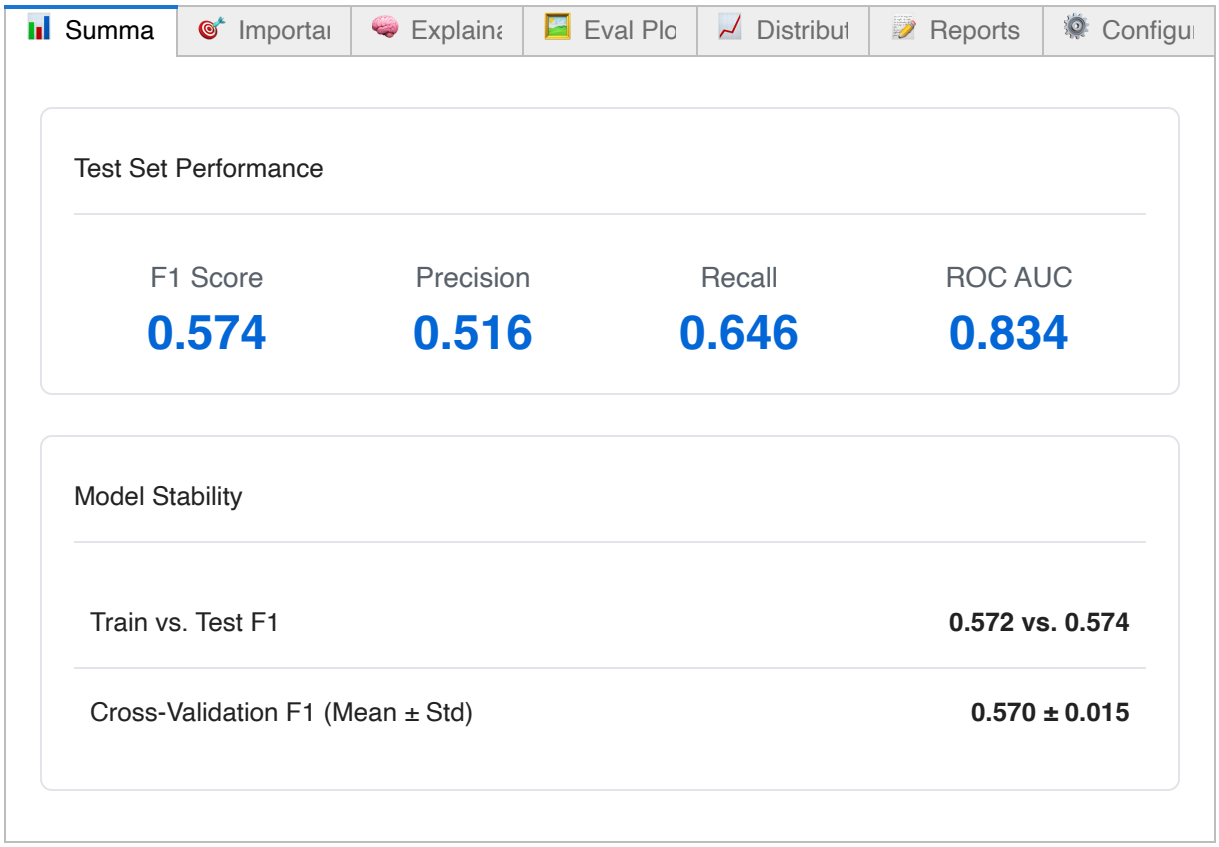
Skewed Features

	skewness
time_spend_company	1.924189
Work_accident	1.922905
promotion_last_5years	6.426524

Outlier Counts

	outlier_count
time_spend_company	4420
Work_accident	7709
promotion_last_5years	1400

Run: gnb_demo_01 | ✅ **F1 (Test):** 0.574



⚠️ SHAP Error Handling

▼ click to expand section

Some models — such as `SVC`, `SVR`, or `GaussianNB` — do not expose traditional feature importance attributes or are incompatible with SHAP explainability tools.

This suite handles such situations **gracefully**:

- The SHAP tab will be skipped silently if no compatible features are found.
- A warning will be logged (but not treated as a failure).
- All other evaluation plots and metrics will still render normally.

This ensures that the workflow remains robust even for models with limited explainability tooling.

In [5]:
```
# ========== Logistic Regression ==========
run_experiment(user_config_path="config/classifier/logreg.yaml")
```

```
Registered model 'logreg_baseline_01' already exists. Creating a new version
of this model...
Created version '4' of model 'logreg_baseline_01'.
```

```
✅ Run complete: `logreg_baseline_01`
📁 Artifacts saved to:
   - Plots:   exports/plots/logreg_baseline_01
   - Reports: exports/reports/logreg_baseline_01
📦 MLflow model: `logreg_baseline_01`

--- Rendering Dashboards ---
```

☐ 🔍 Pre-Modeling Diagnostics

Overview | Missingness | Collinearity | Distributions | Eval Plots

Diagnostics Overview

- **3** numeric features are highly skewed.
- **3** numeric features have potential outliers.
- **3** features have high multicollinearity (VIF > 10.0).

Run: logreg_baseline_01 | ✅ **F1 (Test):** 0.686

Summ | ⚔ Baseli | Import | Explai | Eval F | Distrib | Repor | Config

Comparison to Baseline (gnb_demo_01)

Metric	Current Model	Baseline (gnb_demo_01)	Change
F1	0.686	0.574	+0.113
Precision	0.569	0.516	+0.053
Recall	0.865	0.646	+0.219
Roc Auc	0.916	0.834	+0.082

In [6]:
```
# ========== XGBoost Classifier ==========
```

```
run_experiment(user_config_path="config/classifier/xgboost.yaml")
```

```
Fitting 5 folds for each of 108 candidates, totalling 540 fits
Registered model 'xgb_demo_01' already exists. Creating a new version of thi
s model...
Created version '4' of model 'xgb_demo_01'.
✅ Run complete: `xgb_demo_01`
🗃️ Artifacts saved to:
   - Plots:   exports/plots/xgb_demo_01
   - Reports: exports/reports/xgb_demo_01
📦 MLflow model: `xgb_demo_01`

--- Rendering Dashboards ---
```

☐🔍 Pre-Modeling Diagnostics

📋 Overview | 🧩 Missingness | 🔁 Collinearity | 📊 Distributions | 🖼️ Eval Plots

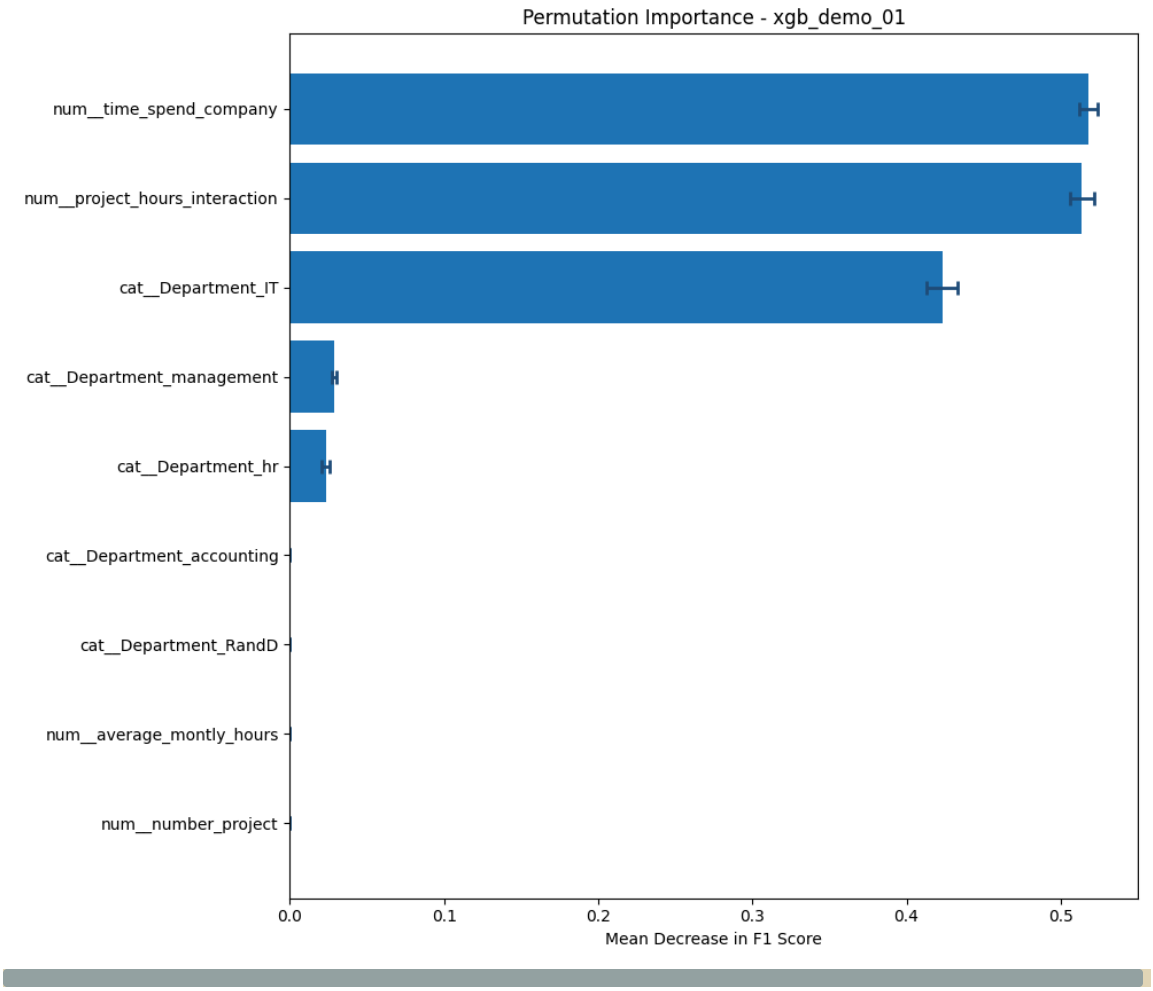
VIF Scores

	feature	VIF
0	last_evaluation	18.815598
1	average_montly_hours	18.091514
2	number_project	13.411024
3	satisfaction_level	6.689082
4	time_spend_company	6.150695
5	Work_accident	1.184356
6	promotion_last_5years	1.030955

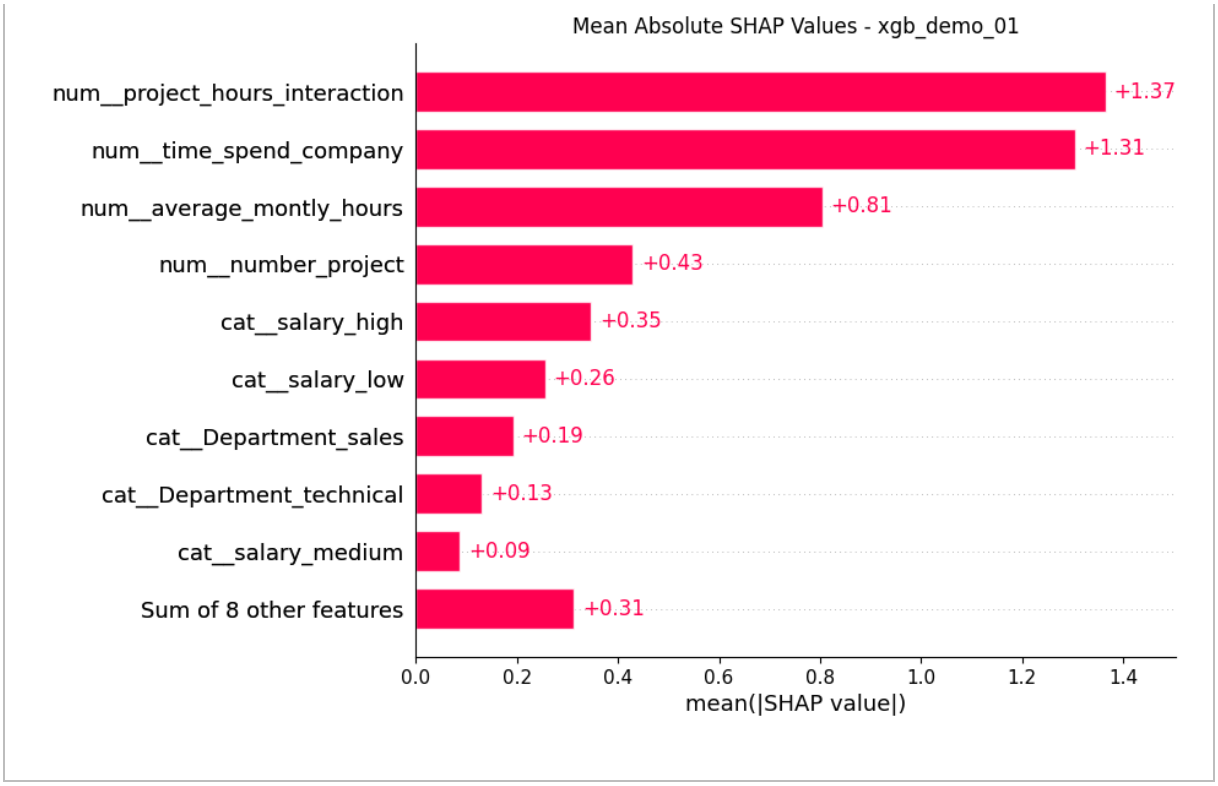
Run: xgb_demo_01 | ✅ **F1 (Test):** 0.929

Summ | Baseli | Import | Explai | Eval F | Distrib | Repor | Config

Permutation Importance



Shap Bar



🔁 Cross-Validation Insight

▼ click here to expand

If hyperparameter tuning via cross-validation is enabled in the config (`hyperparameter_tuning.run: true`), the dashboard will include an additional boxplot in the **Summary** tab.

This plot visualizes the distribution of CV scores across folds for the best-performing parameter set, offering a quick diagnostic of stability and performance variance.

In [7]:
```
# ========== Linear Regression ==========
run_experiment(user_config_path="config/regressor/linreg.yaml")
```

```
Registered model 'linreg_demo_01' already exists. Creating a new version of
this model...
Created version '4' of model 'linreg_demo_01'.
✅ Run complete: `linreg_demo_01`
🗂️ Artifacts saved to:
   - Plots:   exports/plots/linreg_demo_01
   - Reports: exports/reports/linreg_demo_01
📦 MLflow model: `linreg_demo_01`

--- Rendering Dashboards ---
```

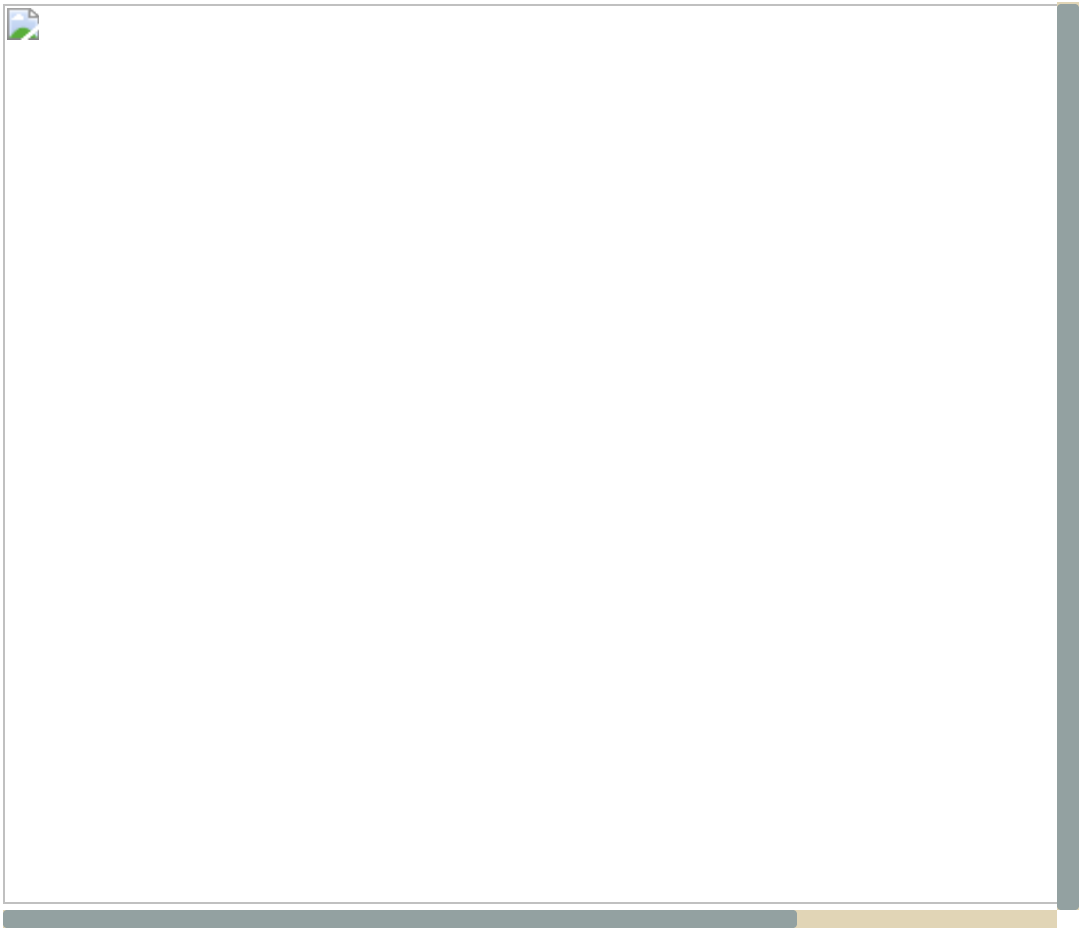
Pre-Modeling Diagnostics

Overview | Missingness | Collinearity | Distributions | Eval Plots

Diagnostic Plots

Plot 1: correlation_matrix.png

Comparison Mode



Run: linreg_demo_01 | ✅ **R-squared:** 0.331

Audit Alerts:

- ⚠️ Weak Fit: R-squared is low (0.331), indicating poor model fit.

Summa | Importa | Explaina | Eval Plo | Distribut | Reports | Configu

Statistical Summary

OLS Regression Results

==

Dep. Variable: left R-squared: 0.336

Model: OLS Adj. R-squared: 0.336

Method: Least Squares F-statistic: 950.0

Date: Tue, 05 Aug 2025 Prob (F-statistic): 0.00

Time: 20:52:48 Log-Likelihood: -7042.4

No. Observations: 30000 AIC: 1.412e+04

Df Residuals: 29983 BIC: 1.426e+04

Df Model: 16

Covariance Type: nonrobust

==

coef std err t P>|t| [0.025 0.975]

---

const -9.38e+09 8.47e+10 -0.111 0.912 -1.75e+11 1.57e+11

num__number_project -0.7656 0.007 -111.520 0.000 -0.779 -0.752

num__average_montly_hours -0.5822 0.006 -103.817 0.000 -0.593 -0.571

```
num__time_spend_company          0.0440     0.002     24.221      0.000
0.040       0.048
num__project_hours_interaction     1.1377     0.010    114.724      0.000
1.118       1.157
cat__Department_IT           -1.596e+09   1.44e+10    -0.111      0.912   -2.
99e+10    2.67e+10
cat__Department_RandD        -1.596e+09   1.44e+10    -0.111      0.912
-2.99e+10    2.67e+10
cat__Department_accounting    -1.596e+09   1.44e+10    -0.111      0.912
-2.99e+10    2.67e+10
cat__Department_hr           -1.596e+09   1.44e+10    -0.111      0.912   -2.
99e+10    2.67e+10
cat__Department_management    -1.596e+09   1.44e+10    -0.111      0.91
2   -2.99e+10    2.67e+10
cat__Department_marketing     -1.596e+09   1.44e+10    -0.111      0.912
-2.99e+10    2.67e+10
cat__Department_product_mng   -1.596e+09   1.44e+10    -0.111      0.91
2   -2.99e+10    2.67e+10
cat__Department_sales        -1.596e+09   1.44e+10    -0.111      0.912   -
2.99e+10    2.67e+10
cat__Department_support      -1.596e+09   1.44e+10    -0.111      0.912   -
2.99e+10    2.67e+10
cat__Department_technical     -1.596e+09   1.44e+10    -0.111      0.912   -
2.99e+10    2.67e+10
cat__salary_high             1.098e+10   9.92e+10     0.111      0.912   -1.83
e+11    2.05e+11
cat__salary_low              1.098e+10   9.92e+10     0.111      0.912   -1.83
e+11    2.05e+11
cat__salary_medium           1.098e+10   9.92e+10     0.111      0.912   -1.
83e+11    2.05e+11
==============================================================
==========================
Omnibus:                    5300.508   Durbin-Watson:                   2.016
Prob(Omnibus):                 0.000   Jarque-Bera (JB):             9069.694
```

```
Skew:                  1.163   Prob(JB):              0.00
Kurtosis:              4.360   Cond. No.           1.50e+15
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The smallest eigenvalue is 3.18e-26. This might indicate that there are strong multicollinearity problems or that the design matrix is singular.
```

🚨 Automated Alert Auditing

▼ click here to expand section

The validation dashboard includes an **Alerts** tab that surfaces automated audit checks on your model's performance.

These alerts are designed to flag potential concerns such as:

- Very low precision or recall
- High class imbalance
- Overfitting indicators (e.g., large delta between train/test scores)
- Underwhelming performance against a baseline (if provided)

This system provides a lightweight, interpretable review of model quality without requiring custom code or manual thresholding.

In [8]:
```
# ========== XGBoost Regressor ==========
run_experiment(user_config_path="config/regressor/xgboost_reg.yaml")
```

```
Fitting 5 folds for each of 108 candidates, totalling 540 fits
Registered model 'xgbreg_demo_01' already exists. Creating a new version of this model...
Created version '4' of model 'xgbreg_demo_01'.
✅ Run complete: `xgbreg_demo_01`
📁 Artifacts saved to:
   - Plots:   exports/plots/xgbreg_demo_01
   - Reports: exports/reports/xgbreg_demo_01
📦 MLflow model: `xgbreg_demo_01`

--- Rendering Dashboards ---
```

Pre-Modeling Diagnostics

Overview | Missingness | Collinearity | Distributions | Eval Plots

Skewed Features

	skewness
time_spend_company	1.924189
Work_accident	1.922905
promotion_last_5years	6.426524

Outlier Counts

	outlier_count
time_spend_company	4420
Work_accident	7709
promotion_last_5years	1400

Run: xgbreg_demo_01 | ✅ **R-squared:** 0.854

Summ | Baseli | Import | Explai | Eval F | Distrib | Repor | Config


```
{
 "run_id": "xgbreg_demo_01",
 "task_type": "regression",
 "pre_model_diagnostics": {
  "run": true,
  "export_reports": true,
  "export_html_report": true,
  "vif_threshold": 10.0,
  "skewness_threshold": 0.75
 },
 "notebook_mode": true,
 "logging": "auto",
 "paths": {
  "input_data": "data/input_data/salifort_50k.csv",
  "reports_dir": "exports/reports",
  "plots_dir": "exports/plots",
  "model_export_dir": "models",
  "metrics_log": "exports/reports/model_metrics_log.csv",
  "log_dir": "logs",
  "train_data_path": "data/dev_data/train_data.csv",
  "test_data_path": "data/dev_data/test_data.csv"
 },
 "modeling": {
  "target_column": "left",
  "test_size": 0.2,
  "feature_engineering": {
   "run": true,
   "module": "model_eval_suite.modeling.feature_engineering",
   "class_name": "HREngineer"
```

```
},
"hyperparameter_tuning": {
  "run": true,
  "param_grid": {
    "estimator__n_estimators": [
      50,
      100,
      150
    ],
    "estimator__max_depth": [
      3,
      5,
      7
    ],
    "estimator__learning_rate": [
      0.01,
      0.1,
      0.2
    ],
    "estimator__subsample": [
      0.8,
      1.0
    ],
    "estimator__colsample_bytree": [
      0.8,
      1.0
    ]
  },
  "cv_folds": 5,
  "scoring": "r2",
  "verbose": 1
},
"pipeline_factory": {
  "name": "XGBRegressor",
```

```
    "registered_name": "xgbreg_demo_01",
    "numeric_features": [
      "number_project",
      "average_montly_hours",
      "time_spend_company",
      "project_hours_interaction"
    ],
    "categorical_features": [
      "Department",
      "salary"
    ],
    "params": {
      "n_estimators": 100,
      "max_depth": 5,
      "learning_rate": 0.1,
      "objective": "reg:squarederror",
      "random_state": 42
    }
  }
},
"evaluation": {
  "run": true,
  "export_xlsx_summary": false,
  "export_html_dashboard": true,
  "compare_to_baseline": "linreg_demo_01",
  "plots": {
    "confusion_matrix": {
      "save": true
    },
    "roc_curve": {
      "save": true
    },
    "pr_curve": {
      "save": true
```

```
  },
  "learning_curve": {
    "save": true
  },
  "calibration_curve": {
    "save": true
  },
  "threshold_plot": {
    "save": true
  },
  "permutation_importance": {
    "save": true
  },
  "lift_curve": {
    "save": true
  },
  "cumulative_gain": {
    "save": true
  },
  "shap_summary": {
    "save": true
  },
  "feature_distributions": {
    "save": true
  },
  "predicted_vs_actual": {
    "save": true
  },
  "residuals_plot": {
    "save": true
  },
  "residuals_histogram": {
    "save": true
  },
```

```
    "qq_plot": {
      "save": true
    }
  },
  "explainability": {
    "run_shap": true,
    "shap_sample_size": 2000
  },
  "audits": {
    "r_squared_threshold": 0.6,
    "rmse_threshold": 50000.0,
    "accuracy_threshold": 0.8,
    "f1_score_threshold": 0.75
  }
 }
}
```

🏆 Champion Model Validation

This section evaluates a registered MLflow model (your *champion*) against a holdout dataset using a dedicated validation YAML configuration.

Key Features

- Uses its own standalone YAML file (separate from training experiments)
- Accepts an optional **baseline model** for drift detection or performance benchmarking
- Automatically generates:
  - Confidence interval plots (if applicable)
  - Baseline comparison deltas (if a baseline model is provided)
  - Alert audits for performance degradation or instability
- Produces a complete interactive dashboard with:
  - Summary metrics and cross-validation visualizations
  - Explainability and feature importance plots
  - Distribution visualizations for target and predictions
  - Full configuration and environment metadata
- **Tags** the evaluated model in the MLflow Registry using your specified `production_tag`

📍 This workflow is ideal for pre-deployment validation, regression testing, and model promotion decisions.

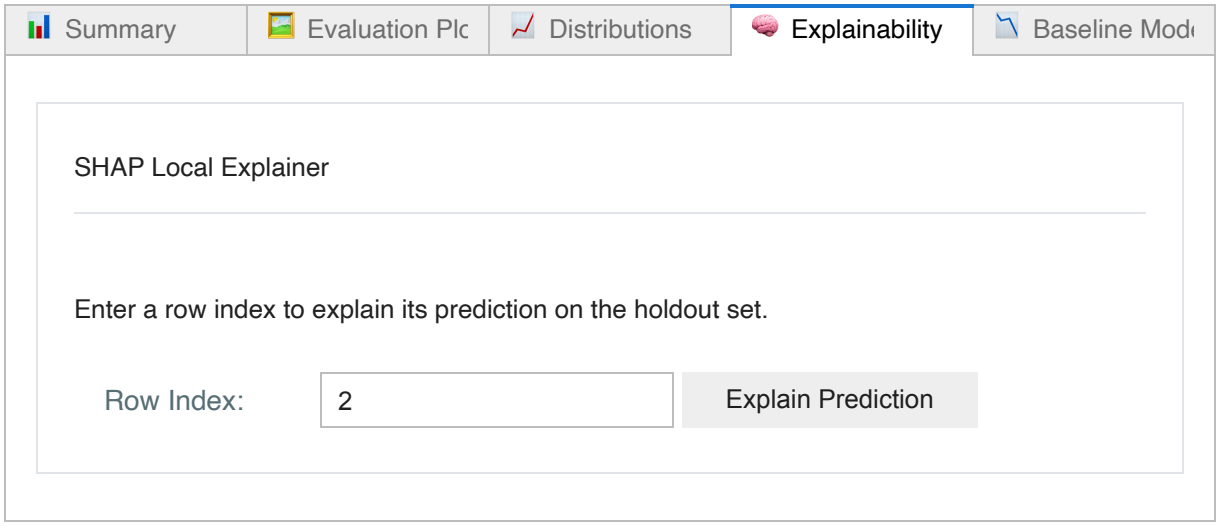
Validation Configurations Used in This Demo

- config/xgb_validation.yaml – XGBoost classifier
- config/xgb_reg_validation.yaml – XGBoost regressor

In [9]:
```
# ========== Validate Classifer Champion Model ==========
validate_champion(config_path="config/xgb_validation.yaml")
```

```
--- 🚀 Starting Champion Model Validation: xgb_demo_01_production_validation
---
Loading model from: models:/xgb_demo_01/1
Loading holdout data from: data/holdout_data/holdout_data.csv
Loading baseline model from: models:/logreg_baseline_01/1
Evaluating baseline model...
Detected task type: classification
Generating final assessment plots...
Exporting validation artifacts...
Tagging model version with status: 'Production-Candidate'
--- ✅ Validation Complete for xgb_demo_01 v1 ---
--- 📊 Rendering Validation Dashboard ---
```

Champion Model Validation: xgb_demo_01_production_validation



In [10]:
```
# ========== Validate Regressor Champion Model ==========
validate_champion(config_path="config/xgb_reg_validation.yaml")
```

```
--- 🚀 Starting Champion Model Validation: xgbreg_demo_01_production_validat
ion ---
Loading model from: models:/xgbreg_demo_01/1
Loading holdout data from: data/holdout_data/holdout_data.csv
Loading baseline model from: models:/linreg_demo_01/1
Evaluating baseline model...
```

```
Detected task type: regression
Generating final assessment plots...
⚠️ Prediction interval plot is only available for ensemble models like Rando
mForestRegressor.
Exporting validation artifacts...
Tagging model version with status: 'Production-Candidate'
--- ✅ Validation Complete for xgbreg_demo_01 v1 ---
--- 📊 Rendering Validation Dashboard ---
```

Champion Model Validation: xgbreg_demo_01_production_validation

Summary | Evaluation Plc | Distributions | Explainability | Baseline Mod

Baseline Model Metrics

r_squared	0.326
mean_absolute_error	0.225
mean_squared_error	0.095
root_mean_squared_error	0.308

✅ Wrap-Up and Next Steps

You've now run multiple models through the full suite — from preprocessing and diagnostics to evaluation and champion validation.

This notebook demonstrates the flexibility of the system, including:

- YAML-driven configuration at every stage
- Reusable pipelines for both classification and regression tasks
- Support for custom feature engineering and hyperparameter tuning
- Interactive dashboards for diagnostics and final reporting
- MLflow integration for model tracking and registry updates

Next Steps

- **Test additional models** by duplicating a config YAML.
- **Customize features** using your own transformers or fe_config modules.

- **Enable advanced diagnostics**, SHAP, and permutation importance as needed.
- **Package and deploy** validated models via the MLflow registry.

For more examples and config templates, explore the `config_resources/` folder or the full README.md.

> Questions or suggestions? Feel free to submit an issue or feature request in the repository.