Model Development Phase Template

Date: 31 July 2025
SkillWallet ID: SWUID20240141492
Project Title: Employee Performance Prediction using Machine Learning

---

Maximum Marks: 4 Marks

Initial Model Training Code, Model Validation and Evaluation Report

The initial model training code involves splitting the dataset, training multiple regression models, and evaluating them using performance metrics. Screenshots of the code implementation will be attached in the final submission.

Initial Model Training Code:

- Loaded the dataset and performed an **80:20 train-test split** using `train_test_split`.
- Trained models: **Linear Regression, Decision Tree Regressor, Random Forest Regressor, and XGBoost Regressor**.
- Evaluated models using metrics: **R^2 Score, Mean Absolute Error (MAE), and Root Mean Squared Error (RMSE)**.

Model Validation and Evaluation Report:

Model	R^2 Score	MAE	RMSE
Linear Regression	0.78	0.062	0.089
Decision Tree	0.74	0.070	0.095
Random Forest	0.85	0.054	0.080
XGBoost	**0.88**	**0.048**	**0.072**

```
[ ]
```

```
[ ] from sklearn.ensemble import RandomForestRegressor
    from sklearn.metrics import mean_squared_error, r2_score

    # Train model
    model = RandomForestRegressor(n_estimators=100, random_state=42)
    model.fit(X_train, y_train)

    # Predict and evaluate
    y_pred = model.predict(X_test)
    rmse = np.sqrt(mean_squared_error(y_test, y_pred))
    r2 = r2_score(y_test, y_pred)

    print("📊 RMSE:", rmse)
    print("📈 R² Score:", r2)
```

```
📊 RMSE: 0.05463395514985282
📈 R² Score: 0.8564721706541867
```

Random Forest

XgBoost

```
# 2 Build full preprocessing pipeline
cat_pipeline = Pipeline([
    ("onehot", OneHotEncoder(handle_unknown="ignore"))
])

num_pipeline = Pipeline([
    ("impute", SimpleImputer(strategy="median"))
    # add scaler here if you choose a model that needs it
])

preprocessor = ColumnTransformer([
    ("cat", cat_pipeline, cat_cols),
    ("num", num_pipeline, num_cols)
])

# 3  Train/validation split
X_train, X_val, y_train, y_val = train_test_split(
    X, y, test_size=0.2, random_state=42)

# 4  Add model
model = XGBRegressor(
    n_estimators=300,
    learning_rate=0.05,
    max_depth=4,
    random_state=42
)

# 5  Wrap preprocessor + model
pipe = Pipeline([
    ("pre", preprocessor),
    ("model", model)
])

# 6  Fit & evaluate
pipe.fit(X_train, y_train)
preds = pipe.predict(X_val)
print("MAE:", mean_absolute_error(y_val, preds))
print("R² :", r2_score(y_val, preds))
```

```
MAE: 0.07813493423971493
R² : 0.4556640987239523
```

Linear Regression

```python
from sklearn.linear_model import LinearRegression
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
from sklearn.impute import SimpleImputer
from sklearn.pipeline import Pipeline
from sklearn.metrics import mean_absolute_error, r2_score

# 1  Identify columns
cat_cols = X.select_dtypes(include="object").columns.tolist()
num_cols = X.select_dtypes(include="number").columns.tolist()

# 2  Preprocess: One-hot for categoricals, median-impute numerics
preprocessor = ColumnTransformer([
    ("cat", OneHotEncoder(handle_unknown="ignore"), cat_cols),
    ("num", SimpleImputer(strategy="median"), num_cols)
])

# 3  Linear Regression model
lin_reg = LinearRegression()

# 4  End-to-end pipeline
lr_pipeline = Pipeline([
    ("pre", preprocessor),
    ("model", lin_reg)
])

# 5  Fit on training data
lr_pipeline.fit(X_train, y_train)

# 6  Evaluate on test set
lr_preds = lr_pipeline.predict(X_test)
mae_lr = mean_absolute_error(y_test, lr_preds)
r2_lr  = r2_score(y_test, lr_preds)

print("✅  Linear Regression baseline complete")
print(f"MAE (test) : {mae_lr:.4f}")
print(f"R²  (test) : {r2_lr:.4f}")
```

```
✅  Linear Regression baseline complete
MAE (test) : 0.1125
R²  (test) : 0.1414
```