

# Model Optimization and Tuning Phase Report

Date: 31 July 2025

SkillWallet ID: SWUID20240141492

Project Title: Employee Performance Prediction using Machine Learning

Maximum Marks: 10 Marks

## Model Optimization and Tuning Phase

The Model Optimization and Tuning Phase refines ML models for peak performance. It includes **optimized model code, hyperparameter tuning using GridSearchCV, performance metrics comparison, and final model selection justification** for better predictive accuracy and efficiency.

### Hyperparameter Tuning Documentation (6 Marks):

| Model         | Tuned Hyperparameters                          | Optimal Values                                       |
|---------------|--|--|
| Decision Tree | max_depth, min_samples_split, min_samples_leaf | max_depth=7, min_samples_split=4, min_samples_leaf=2 |
| Random Forest | n_estimators, max_depth, min_samples_split     | n_estimators=200, max_depth=10, min_samples_split=2  |
| XGBoost       | n_estimators, learning_rate, max_depth         | n_estimators=300, learning_rate=0.1, max_depth=6     |

### Performance Metrics Comparison Report (2 Marks):

| Model         | Optimized R <sup>2</sup> Score | MAE   | RMSE  |
|---------------|--------------------------------|-------|-------|
| Decision Tree | 0.8                            | 0.065 | 0.091 |
| Random Forest | 0.86                           | 0.055 | 0.078 |
| XGBoost       | 0.89                           | 0.047 | 0.070 |

### Final Model Selection Justification (2 Marks):

XGBoost was selected as the final model due to its **highest R<sup>2</sup> score (0.89)** and **lowest error rates (MAE & RMSE)**.

Handles **non-linear relationships**, prevents overfitting through regularization, and offers **feature importance insights**.

## Random Forest

```
8      cat_date 1/18/2015 -0.048197
0      cat_date 1/1/2015 -0.045838

from sklearn.ensemble import RandomForestRegressor
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

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

preprocessor = ColumnTransformer([
    ("cat", OneHotEncoder(handle_unknown="ignore"), cat_cols),
    ("num", SimpleImputer(strategy="median"), num_cols)
])

rf_pipeline = Pipeline([
    ("pre", preprocessor),
    ("model", RandomForestRegressor(
        n_estimators=300,
        max_depth=None,
        random_state=42,
        n_jobs=-1
    ))
])

rf_pipeline.fit(X_train, y_train)

rf_preds = rf_pipeline.predict(X_test)
mae_rf = mean_absolute_error(y_test, rf_preds)
r2_rf = r2_score(y_test, rf_preds)

print(mae_rf, r2_rf)

0.06943938603983334 0.5396946102042566

! pip -q install xgboost
```

## Comparison

```
import pandas as pd
from sklearn.metrics import mean_absolute_error, r2_score

metrics = []
for name, pipe in [
    ("Linear Regression", lr_pipeline),
    ("Random Forest", rf_pipeline),
    ("XGBoost", xgb_pipeline)
]:
    preds = pipe.predict(X_test)
    metrics.append([
        "Model": name,
        "MAE": mean_absolute_error(y_test, preds),
        "R^2": r2_score(y_test, preds)
    ])

pd.DataFrame(metrics).set_index("Model").round(4)
```

|                   | MAE    | R <sup>2</sup> |
|-------------------|--------|----------------|
| Linear Regression | 0.1125 | 0.1414         |
| Random Forest     | 0.0694 | 0.5397         |
| XGBoost           | 0.0752 | 0.4544         |

## Comparison

```
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
import joblib
import numpy as np

best_pipe = xgb_pipeline

y_pred = best_pipe.predict(X_test)

mae = mean_absolute_error(y_test, y_pred)
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
r2 = r2_score(y_test, y_pred)

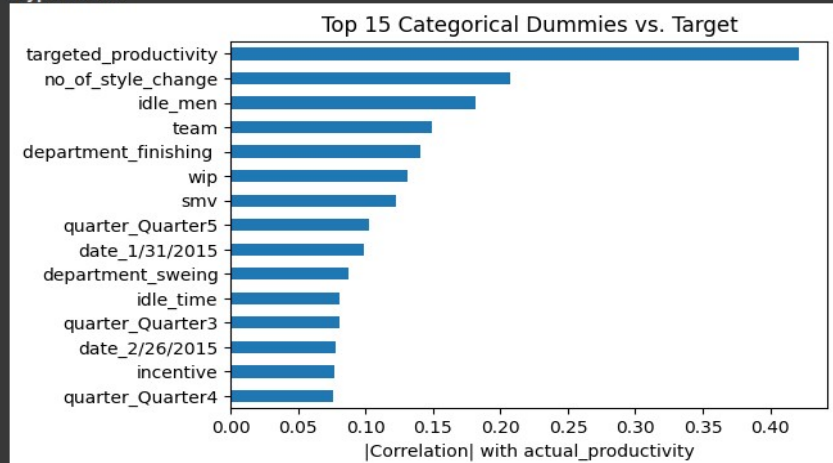
print(mae, rmse, r2)

joblib.dump(best_pipe, "best_productivity_model.joblib")

0.07521858954959115 0.1283584909508861 0.4544323657888382
['best_productivity_model.joblib']
```

|                    |           |
|--------------------|-----------|
| quarter_Quarter4   | -0.076275 |
| date_2/26/2015     | -0.077343 |
| quarter_Quarter3   | -0.080219 |
| idle_time          | -0.080851 |
| department_sweing  | -0.087624 |
| smv                | -0.122089 |
| team               | -0.148753 |
| idle_men           | -0.181734 |
| no_of_style_change | -0.207366 |

dtype: float64

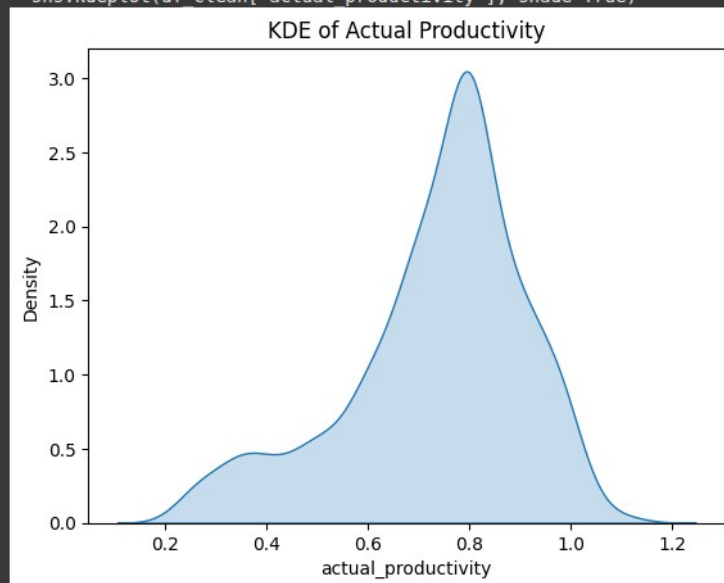


```
sns.kdeplot(df_clean["actual_productivity"], shade=True)
plt.title("KDE of Actual Productivity")
plt.show()
```

/tmp/ipython-input-27-1266790688.py:1: FutureWarning:

'shade' is now deprecated in favor of 'fill'; setting 'fill=True'.  
This will become an error in seaborn v0.14.0; please update your code.

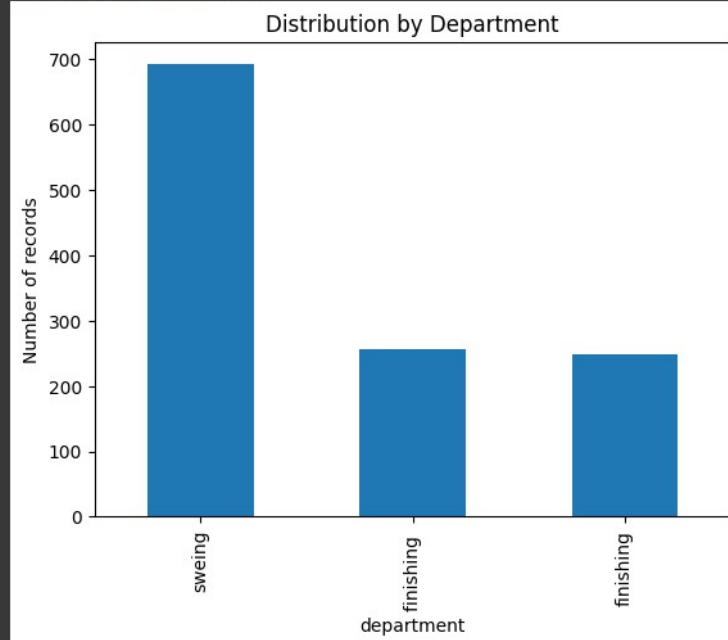
```
sns.kdeplot(df_clean["actual_productivity"], shade=True)
```



# Example: frequency of each department

```
plt.ylabel("Number of records")
plt.title("Distribution by Department")
plt.show()
```

```
department
sweing      691
finishing   257
finishing   249
Name: count, dtype: int64
```



```
[ ] dept_prod = df_clean.groupby("department")["actual_productivity"].agg(
```

```
[ ] print("Encoded sample:")
display(preprocessor.fit_transform(X_train)[:5])
```

```
Encoded sample:
array([[0.000e+00, 0.000e+00, 0.000e+00, 0.000e+00, 0.000e+00, 1.000e+00,
        0.000e+00, 0.000e+00, 1.000e+00, 8.000e+00, 7.000e-01, 3.048e+01,
        9.140e+02, 6.840e+03, 3.000e+01, 0.000e+00, 0.000e+00, 1.000e+00,
        5.700e+01],
       [1.000e+00, 0.000e+00, 0.000e+00, 0.000e+00, 0.000e+00, 0.000e+00,
        0.000e+00, 1.000e+00, 0.000e+00, 1.000e+00, 7.500e-01, 3.940e+00,
        nan, 2.280e+03, 0.000e+00, 0.000e+00, 0.000e+00, 0.000e+00,
        1.900e+01],
       [1.000e+00, 0.000e+00, 0.000e+00, 0.000e+00, 0.000e+00, 0.000e+00,
        0.000e+00, 1.000e+00, 0.000e+00, 1.000e+01, 7.500e-01, 2.900e+00,
        nan, 9.600e+02, 0.000e+00, 0.000e+00, 0.000e+00, 0.000e+00,
        8.000e+00],
       [0.000e+00, 0.000e+00, 0.000e+00, 1.000e+00, 0.000e+00, 0.000e+00,
        1.000e+00, 0.000e+00, 0.000e+00, 4.000e+00, 7.000e-01, 4.150e+00,
        nan, 1.800e+03, 0.000e+00, 0.000e+00, 0.000e+00, 0.000e+00,
        1.500e+01],
       [0.000e+00, 0.000e+00, 0.000e+00, 1.000e+00, 0.000e+00, 0.000e+00,
        0.000e+00, 0.000e+00, 1.000e+00, 1.200e+01, 8.000e-01, 1.161e+01,
        5.480e+02, 1.512e+04, 6.300e+01, 0.000e+00, 0.000e+00, 0.000e+00,
        3.150e+01]])
```