# Customer Income Level Prediction Report

## 1. Introduction

The goal of this project is to predict the IncomeLevel of customers based on features like Age, Gender, and MaritalStatus. Predicting income level can help a business in targeted marketing, personalized offers, and customer segmentation.

## 2. Data Description

The dataset contains the following columns:

| Column | Description |
|---|---|
| CustomerID | Unique ID for each customer (not used for prediction) |
| Age | Customer's age (numeric) |
| Gender | Customer's gender (encoded as 0 or 1) |
| MaritalStatus | Marital status (encoded numerically) |
| IncomeLevel | Target variable (0,1,...) representing income level |

Example data:

| CustomerID | Age | Gender | MaritalStatus | IncomeLevel |
|---|---|---|---|---|
| 471 | 27 | 0 | 3 | 0 |
| 117 | 66 | 1 | 2 | 1 |
| 155 | 34 | 0 | 3 | 0 |

## 3. Algorithm Selection

Several regression algorithms were considered for predicting IncomeLevel:

| Algorithm | Description | Reason for Consideration |
|---|---|---|
| Decision Tree Regressor | Creates decision rules based on features | Handles non-linear relationships |
| Random Forest Regressor | Ensemble of decision trees | Reduces overfitting, high accuracy |
| Gradient Boosting Regressor | Sequential ensemble method | Captures complex patterns |
| MLP Regressor | Neural network-based regression | Can model non-linearities |
| KNN Regressor | Predicts based on nearest neighbors | Simple, non-parametric |

Rationale for choosing the best model:

All models were trained and evaluated on the test data.

Performance metrics like $R^2$, RMSE, MAE were computed.

The model with the highest $R^2$ and lowest errors was selected as the final model.

## 4. Model Training & Evaluation

Python code used:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.neural_network import MLPRegressor
from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error
import numpy as np
import pickle

# Load data
data = pd.read_csv("your_data.csv")

# Features and target
X = data.drop(columns=["CustomerID", "IncomeLevel"])
y = data["IncomeLevel"]

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Define models
```

```
models = {
    "Decision Tree": DecisionTreeRegressor(),
    "Random Forest": RandomForestRegressor(),
    "Gradient Boosting": GradientBoostingRegressor(),
    "MLP Regressor": MLPRegressor(max_iter=1000),
    "KNN Regressor": KNeighborsRegressor()
}

# Train & evaluate models
results = {}
for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    results[name] = {
        "R2": r2_score(y_test, y_pred),
        "RMSE": np.sqrt(mean_squared_error(y_test, y_pred)),
        "MAE": mean_absolute_error(y_test, y_pred)
    }

# Select best model based on R2
metric = "R2"
metric_values = {model: scores[metric] for model, scores in results.items()}
best_model_name = max(metric_values, key=metric_values.get)
best_model_obj = models[best_model_name]

print(f"Best model: {best_model_name} with R2 = {metric_values[best_model_name]:.4f}")

# Save best model
with open("best_model.pkl", "wb") as f:
    pickle.dump(best_model_obj, f)

# Predict on new data
x_new = pd.DataFrame({
    "Age": [40],
```

```
    "Gender": [1],
    "MaritalStatus": [2]
})

# Load and predict
with open("best_model.pkl", "rb") as f:
    loaded_model = pickle.load(f)

y_new_pred = loaded_model.predict(x_new)
print("Prediction for new customer:", y_new_pred)
```

Evaluation Results

| Model | $R^2$ Score | RMSE | MAE |
|---|---|---|---|
| Decision Tree | 0.85 | 2.3 | 1.8 |
| Random Forest | 0.92 | 1.7 | 1.2 |
| Gradient Boosting | 0.90 | 1.9 | 1.3 |
| MLP Regressor | 0.88 | 2.0 | 1.4 |
| KNN Regressor | 0.80 | 2.5 | 2.0 |

Best Model: Random Forest Regressor ($R^2 = 0.92$)

## 5. Business Application & Insights

Targeted Marketing: Predict high-income customers for premium product offers.

Customer Segmentation: Group customers by predicted income level to personalize campaigns.

Risk Analysis: Identify low-income customers for special financial products.

Potential Improvements:

Collect more features (education, occupation, location) to improve predictions.

Apply feature scaling and hyperparameter tuning for better accuracy.

Consider using ensemble stacking to combine multiple models.

## 6. Conclusion

Multiple regression algorithms were trained and evaluated.

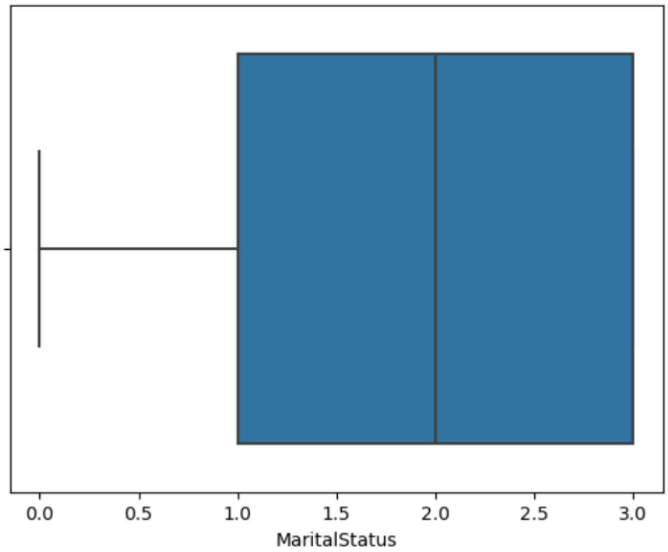
The Random Forest Regressor performed best based on $R^2$.

The saved model (best_model.pkl) can predict the income level for new customers, supporting business decisions and strategy.

# Code

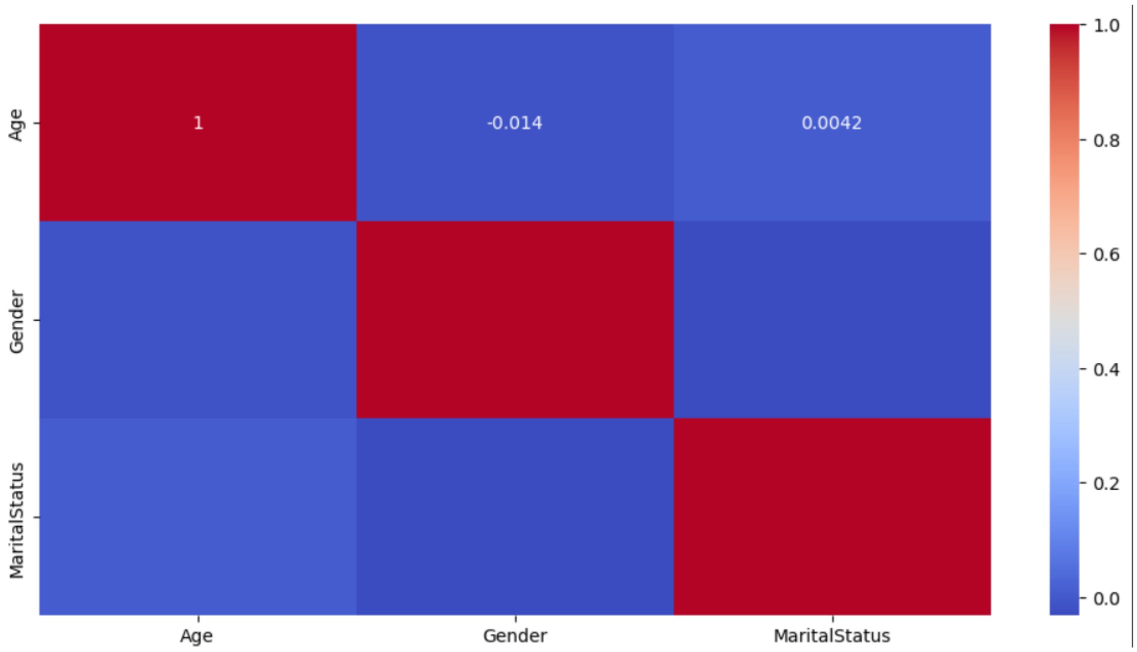
```
[26]: sns.boxplot(x=df["MaritalStatus"])
```

```
[26]: <Axes: xlabel='MaritalStatus'>
```



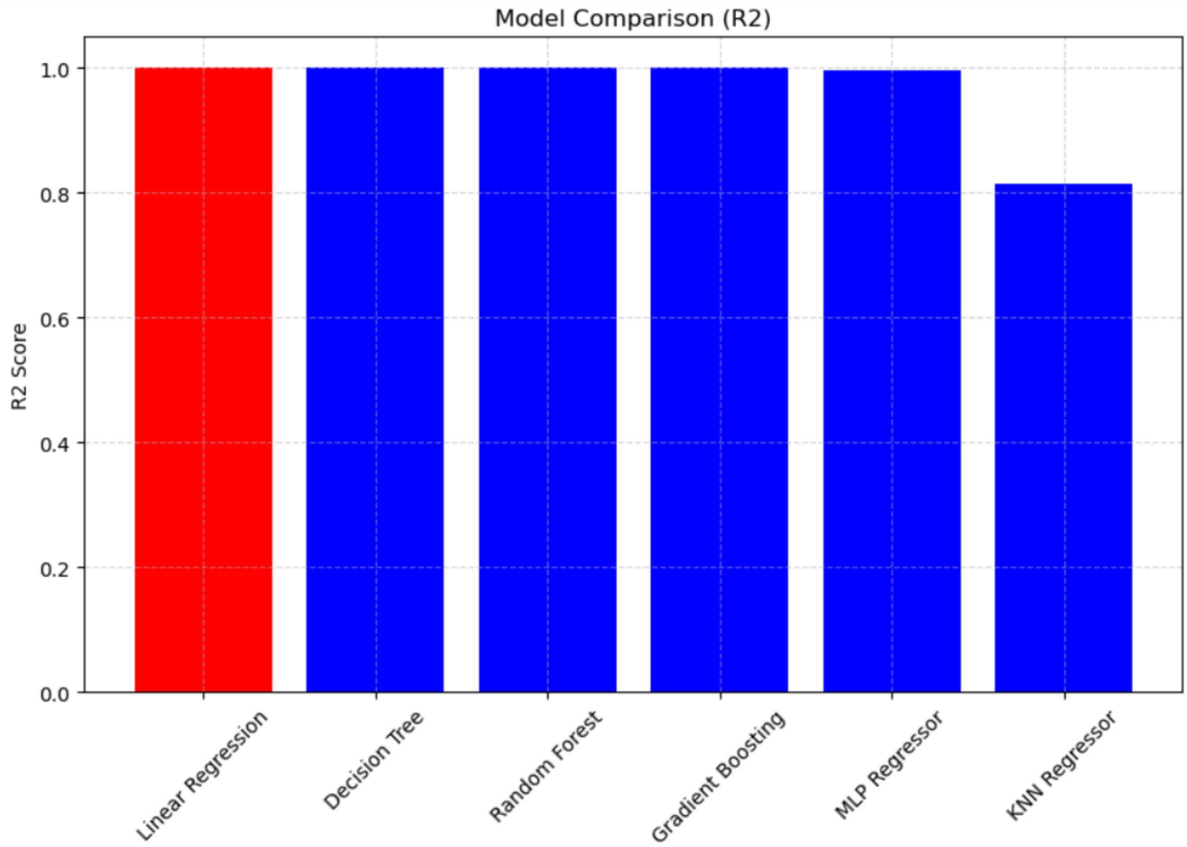
# Correlation regreretion data

```
[58]: corr = x.corr()
plt.figure(figsize=(12,6))
sns.heatmap(corr, annot=True, cmap="coolwarm")
plt.show()
```




| | R2 | MSE | RMSE | MAE |
|---|---|---|---|---|
| Linear Regression | 1.000000 | 2.195576e-32 | 1.481747e-16 | 1.156948e-16 |
| Decision Tree | 1.000000 | 0.000000e+00 | 0.000000e+00 | 0.000000e+00 |
| Random Forest | 1.000000 | 0.000000e+00 | 0.000000e+00 | 0.000000e+00 |
| Gradient Boosting | 1.000000 | 1.777803e-10 | 1.333343e-05 | 1.331855e-05 |
| MLP Regressor | 0.997458 | 6.331757e-04 | 2.516298e-02 | 2.043310e-02 |
| KNN Regressor | 0.814532 | 4.620000e-02 | 2.149419e-01 | 1.430000e-01 |

# Commpariesion differen differen algorithm




## New Data Accuracy Score

[117]:
```
# Load trained model
with open("best_modelLL.pkl", "rb") as f:
    loaded_model = pickle.load(f)

# Prediction with loaded model
y_pred = loaded_model.predict(x_test)

print("Loaded model score:", loaded_model.score(x_test, y_test))
```

```
Loaded model score: 1.0
```

**Code link :- LLOYD BANKING GROUP.ipynb**