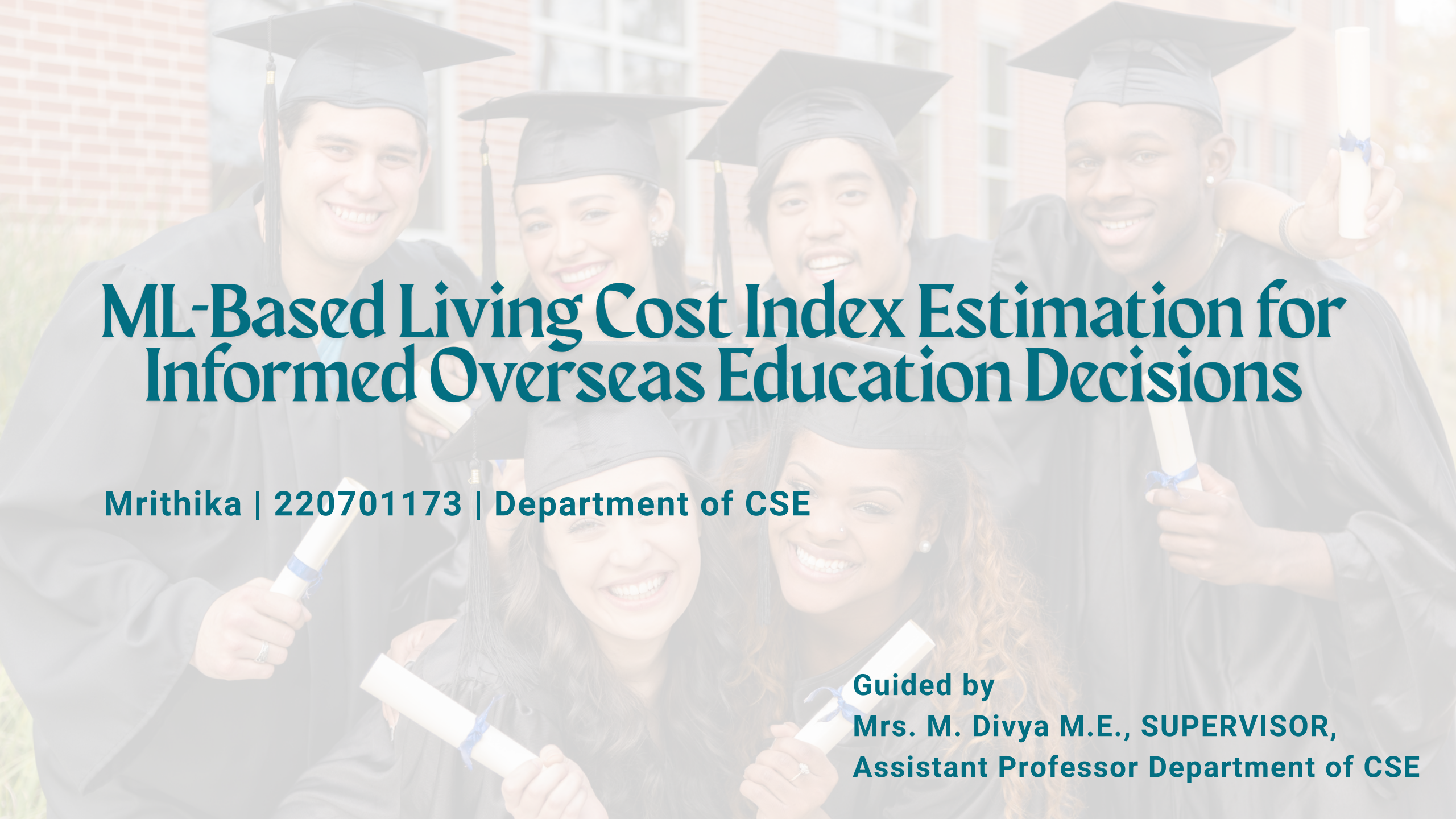


ML-Based Living Cost Index Estimation for Informed Overseas Education Decisions

Mrithika | 220701173 | Department of CSE

Guided by
Mrs. M. Divya M.E., SUPERVISOR,
Assistant Professor Department of CSE

INTRODUCTION

This project aims to assist students planning to study abroad by predicting the Living Cost Index based on key factors like country, city, tuition fees, rent, and more. A Random Forest Regressor model was used due to its robustness, accuracy, and ability to handle both categorical and numerical features effectively. By analyzing past data and estimating the relative cost of living, the model helps users make informed financial decisions when selecting study destinations.

LITERATURE SURVEY

Numbeo is a widely-used platform that provides crowd-sourced cost-of-living data for cities across the globe. It offers users a general idea of monthly expenses in various regions, including rent, groceries, and transportation. However, it does not incorporate educational variables such as tuition fees or program type, making it insufficient for students planning their overseas education budgets.

LITERATURE SURVEY

QS Top Universities, a leading higher education platform, offers estimates for tuition fees and cost of living by location and field of study. While useful for understanding financial ranges at a high level, its static nature and lack of user-driven predictions limit its application for personalized budget planning. Students still need to manually piece together fragmented information.

LITERATURE SURVEY

Research in the Journal of International Students emphasizes the economic pressure on international students and the need for better financial forecasting. These studies often focus on post-arrival experiences and survey-based insights rather than predictive tools. This creates a gap in pre-decision support systems, which machine learning models like Random Forest can fill by offering tailored cost estimations based on multiple dynamic inputs.

OBJECTIVES

- Predict living cost index for studying abroad using ML.
- Use key factors like tuition, rent, and visa fees.
- Build a simple UI for cost estimation.
- Help students plan finances effectively.

SYSTEM ARCHITECTURE

DATA LAYER

Preprocessed dataset with features like country, city, university, program, tuition, rent, insurance, and visa fees.

↓

PREPROCESSING

Label encoding for categorical features, derived features like Annual Rent and Total Cost, and scaling using StandardScaler.

↓

MODEL

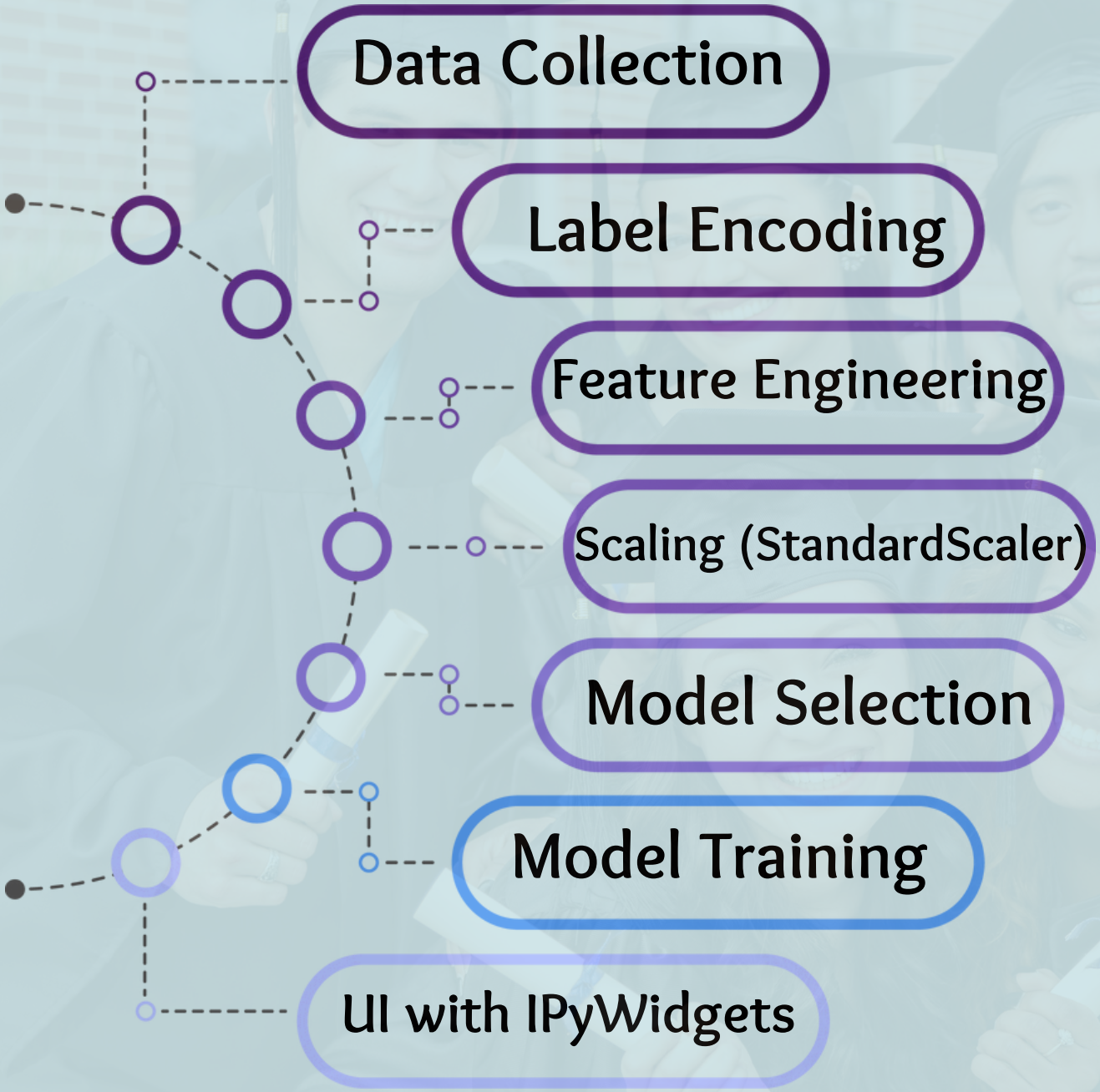
Random Forest Regressor trained to predict the Living Cost Index.

↓

PREDICTION LAYER

Processes user inputs and predicts the Living Cost Index.

METHODOLOGY



Why Random Forest Regressor ?

- Handles mixed data types
- Robust to overfitting
- Captures non-linear patterns
- Minimal preprocessing
- High accuracy on diverse data
- Feature importance insights

METHODOLOGY

- Data Collection: Base for training
- Label Encoding: Categorical → Numeric
- Feature Engineering: Added rent & total cost
- Scaling: Normalized inputs
- Model (RFR): Robust & accurate
- Training: Learn from data
- UI (IPyWidgets): Easy user input & output

IMPLEMENTATION

Data Preprocessing: Handling Missing Values

python

```
# Fill missing values with the median for numerical columns

data['Tuition_USD'].fillna(data['Tuition_USD'].median(), inplace=True)
```

Feature Scaling: Standardization

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
data[['Tuition_USD', 'Living_Cost_Index']] = scaler.fit_transform(data[['Tuition_USD', 'Living_Cost_Index']])
```

IMPLEMENTATION

Splitting Data into Training and Test Sets

```
from sklearn.model_selection import
train_test_split
X = data[['Tuition_USD',
'Living_Cost_Index']]  # Features
y = data['Total_Cost']  # Target variable
X_train, X_test, y_train, y_test =
train_test_split(X, y, test_size=0.2,
random_state=42)
```

Model Training: Random Forest Regressor

```
from sklearn.ensemble import
RandomForestRegressor
model =
RandomForestRegressor(n_estimator
s=100, random_state=42)
model.fit(X_train, y_train)
```

IMPLEMENTATION

Model Hyperparameter Tuning: RandomizedSearchCV

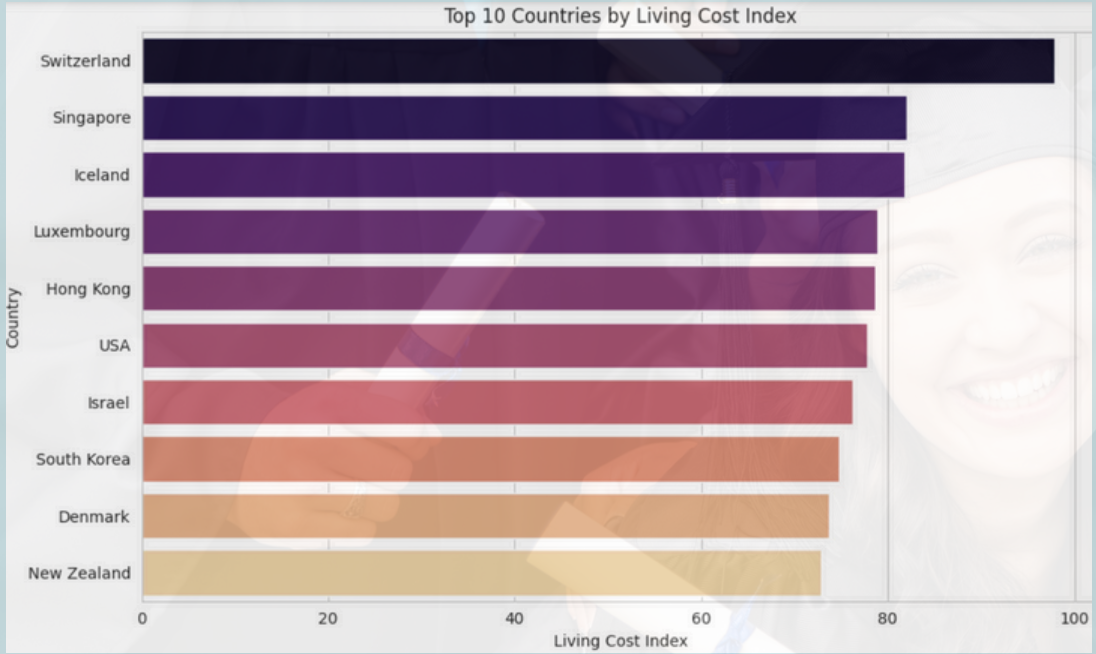
```
from sklearn.model_selection import RandomizedSearchCV
param_dist = {'n_estimators': [50, 100, 200], 'max_depth': [10, 20, None]}
random_search = RandomizedSearchCV(model,
param_distributions=param_dist, n_iter=10, random_state=42)
random_search.fit(X_train, y_train)
print(f"Best Parameters: {random_search.best_params_}")
```

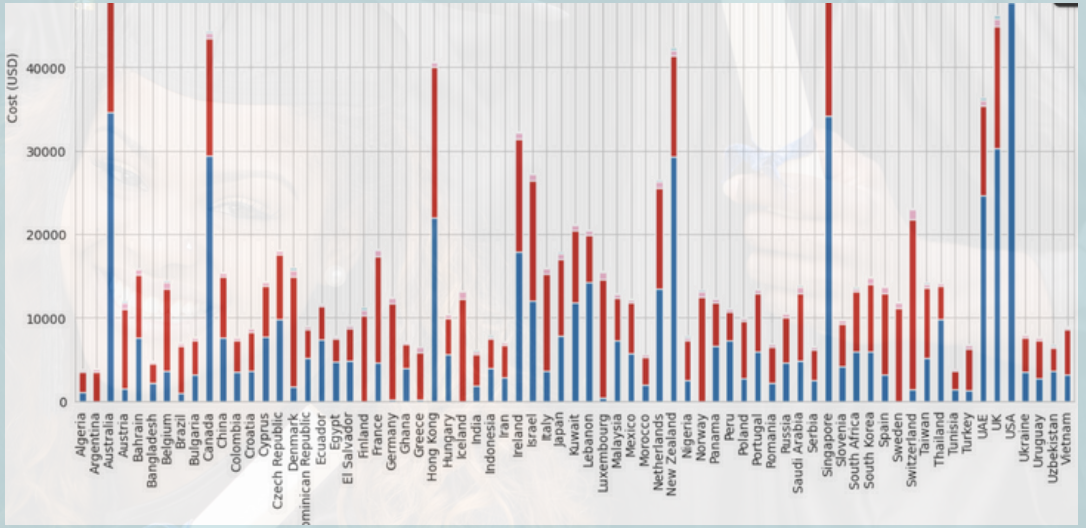
RESULTS

```
[24] r2=r2_score(y_test,y_pred)
     print(f" R2 Score : {r2*100}")
```

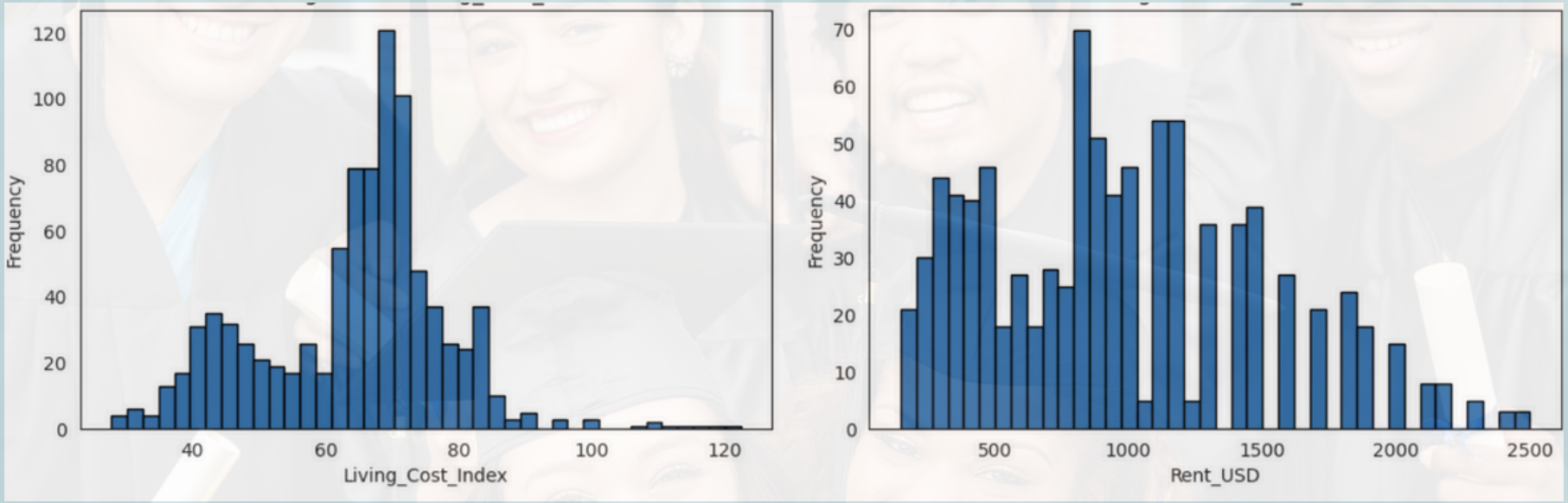
0s

```
R2 Score : 96.04187254075408
```





RESULTS



After completing the Exploratory Data Analysis (EDA), which included data cleaning, preprocessing, and visualization, the machine learning model was trained and evaluated. The model achieved an accuracy of 96.0418%, indicating strong performance in predicting the target variable.

COMPARISON WITH EXISTING WORK

Model/Approach	Accuracy (%)	Description
Our Model (Random Forest)	**96.0418**	Achieved the highest accuracy after training with the given dataset.
Existing Work 1 (Linear Regression)	85.5	A simple linear regression model, which performs lower than our Random Forest model.
Existing Work 2 (Support Vector Machine)	88.3	SVM is a strong model but still falls short compared to our Random Forest.
Existing Work 3 (XGBoost)	93.2	XGBoost shows good performance, but the accuracy is not as high as Random Forest.

CONCLUSION AND FUTURE WORK

Conclusion:

- Built a Random Forest Regressor with an accuracy of 96.0418%.
- Exploratory Data Analysis (EDA) provided key insights into the dataset.
- Compared with models like Linear Regression, SVM, and XGBoost, the Random Forest model outperformed them.

Future Work:

- Hyperparameter tuning for better performance.
- Explore Gradient Boosting and LightGBM for improved results.
- Incorporate additional features and apply feature selection.
- Address data imbalance with techniques like SMOTE.
- Deploy model for real-time predictions.
- Regular model retraining with updated data for long-term accuracy.

REFERENCE

1. Breiman, L. (2001). "Random Forests." Machine Learning, 45(1), 5-32.
2. Vapnik, V. (1995). The Nature of Statistical Learning Theory.
3. Chen, T., & Guestrin, C. (2016). "XGBoost: A Scalable Tree Boosting System." Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.
4. James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). An Introduction to Statistical Learning: With Applications in R.
5. Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). "SMOTE: Synthetic Minority Over-sampling Technique." Journal of Artificial Intelligence Research, 16, 321-357.
6. Pascanu, R., Mikolov, T., & Bengio, Y. (2013). "On the difficulty of training Recurrent Neural Networks." International Conference on Machine Learning (ICML)


THANKYOU