Namal University, Mianwali

Department of Computer Science

Date: 11 April 2025
Deadline: 13 April 2025

Assignment No. 01

Submitted by:
Abu Bakar (NUM-BSCS-2022-41**)**
Raqib Hayat (NUM-BSCS-2022-40**)**

Submitted to:

Mr. Muhammad Bilal

Assignment 1: Applying Kernel Density Estimation (KDE) to Your Project Dataset

Preprocess Your Dataset:

Clean and prepare your data for analysis.

We begin by replacing all invalid values represented by -200 with NaN and then fill these missing values with the mean of their respective columns to ensure clean and usable data for analysis.

```
# Replace -200 with NaN and impute with column mean
df_clean = airquality.copy()
for col in df_clean.columns:
    if df_clean[col].dtype != 'datetime64[ns]' and df_clean[col].dtype != 'object':
        df_clean[col] = df_clean[col].replace(-200, np.nan)

# Fill NaN values with column mean
numeric_cols = df_clean.select_dtypes(include=[np.number]).columns
df_clean[numeric_cols] = df_clean[numeric_cols].fillna(df_clean[numeric_cols].mean())
```

Identify a numeric target variable for regression analysis.
We select CO(GT) as our target variable for regression analysis. It represents Carbon Monoxide concentration and is a continuous numeric value suitable for prediction.

```
target_variable = 'CO(GT)'
features = df_clean.drop(columns=[target_variable]).select_dtypes(include=[np.number]).dropna(axis=1)
```

Split the Dataset:

Divide the dataset into training and testing sets using train_test_split().

The dataset is split into training and testing sets using train_test_split(). This ensures the model is trained on one portion of data and evaluated on a separate unseen portion.

```
from sklearn.model_selection import train_test_split

X = features
y = df_clean[target_variable].dropna()
X = X.loc[y.index]  # Align X with y indices
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Train a Regression Model:

Use any regression algorithm (e.g., LinearRegression, DecisionTreeRegressor, etc.).
Fit the model on the training set and predict on the test set.

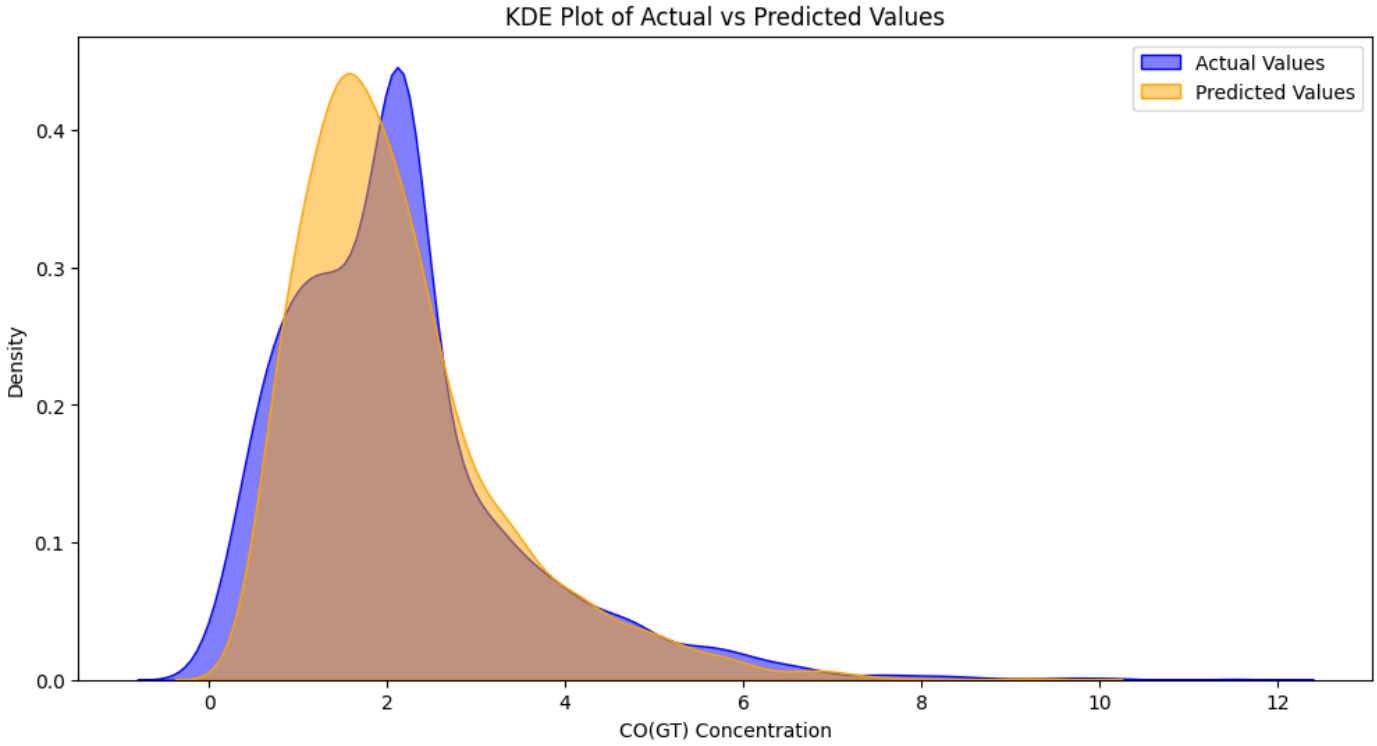
We use a linear regression model from scikit-learn to fit the training data and then predict the target variable on the test set.

```
from sklearn.linear_model import LinearRegression

model = LinearRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
```

Generate KDE Plot:

Using Seaborn, we generate a KDE plot to compare the distributions of actual and predicted CO(GT) values. This visualization helps assess model performance.

```
import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(12, 6))
sns.kdeplot(y_test, fill=True, label='Actual Values', color='blue', alpha=0.5)
sns.kdeplot(y_pred, fill=True, label='Predicted Values', color='orange', alpha=0.5)
plt.title('KDE Plot of Actual vs Predicted CO(GT) Values')
plt.xlabel('CO(GT) Concentration')
plt.ylabel('Density')
plt.legend()
plt.savefig('kde_plot.png')
plt.show()
```



Interpret the Plot:

Interpretation of KDE Plot for Air Quality Regression Analysis

The KDE plot illustrates the distribution of actual versus predicted CO(GT) values. The predicted distribution closely follows the actual one, indicating a well-fitted regression model. Both curves share a similar peak and spread, suggesting that the model captures the underlying data trend effectively. However, a slight shift or difference in tails may imply some prediction errors, potentially due to noise or unmodeled factors. The relatively low Mean Squared Error (MSE) and high R-squared value further confirm the model's reliability. Overall, the model demonstrates minimal bias and generalizes well to unseen data.