**HOUSE PRICE PREDICTION USING CNN**

**SOURCE CODE:**

**import pandas as pd**

**import numpy as np**

**df = pd.read\_csv('/kaggle/input/housing-prices-dataset/Housing.csv')**

**df.head()**

**import seaborn as sns**

**import matplotlib.pyplot as plt**

**plt.figure(figsize=(12, 6))**

**sns.scatterplot(x='area', y='price', hue='stories', data=df,palette="Set2")**

**plt.xlabel('Area')**

**plt.ylabel('Price')**

**plt.title('Price vs Area Scatter Plot')**

**plt.legend(title='Furnishing Status')**

**plt.grid(True)**

**plt.show()**

**fig, axs = plt.subplots(nrows=1, ncols=2, figsize=(14, 6))**

**# Plot the first countplot on the first subplot (axs[0])**

**sns.countplot(x=df["airconditioning"], hue=df["stories"], palette="viridis", ax=axs[0])**

**# Plot the second countplot on the second subplot (axs[1])**

**sns.countplot(x=df["hotwaterheating"], hue=df["stories"], palette="viridis", ax=axs[1])**

**# Optionally, add titles to the subplots**

**axs[0].set\_title("Air conditioning")**

**axs[1].set\_title("Hot water heating")**

**fig, axs = plt.subplots(nrows=2, ncols=3, figsize=(14, 8))**

**# List of column names you want to plot**

**columns = ['price', 'area', 'bedrooms', 'bathrooms', 'stories', 'parking']**

**counter = 0**

**for i in range(2):**

**for j in range(3):**

**sns.boxplot(df[columns[counter]], ax=axs[i, j], palette="viridis")**

**axs[i, j].set\_title(columns[counter].capitalize())**

**counter += 1**

**plt.tight\_layout()**

**plt.show()**

**from sklearn.linear\_model import LinearRegression,Ridge**

**from sklearn.ensemble import RandomForestRegressor,GradientBoostingRegressor**

**from sklearn.svm import SVC**

**models = {**

**'Linear Regression': LinearRegression(n\_jobs=5),**

**'Random Forest': RandomForestRegressor(),**

**'Gradient Boosting':GradientBoostingRegressor(),**

**'Ridge Regression': Ridge(),**

**"SVC":SVC()**

**}**

**for model in models.items():**

**print(model[0])**

**from sklearn.model\_selection import train\_test\_split**

**from sklearn.pipeline import Pipeline**

**from sklearn.preprocessing import StandardScaler**

**from sklearn.model\_selection import GridSearchCV**

**from sklearn.metrics import r2\_score,mean\_absolute\_error,mean\_squared\_error**

**x=df.drop("price",axis=1)**

**y=df["price"]**

**X\_train, X\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size=0.20, random\_state=42)**

**a=1**

**best\_r2=-1**

**for name, model in models.items():**

**pipe = Pipeline([**

**('scaler', StandardScaler()),**

**('model', model)**

**])**

**# Fit the pipeline**

**model=pipe.fit(X\_train, y\_train)**

**y\_pre=model.predict(X\_test)**

**print(f'{a} :- {name} - MSE: {mean\_squared\_error(y\_test, y\_pre):.4f}')**

**print(f' {name} - R2: {r2\_score(y\_test, y\_pre):.4f}')**

**print(f' {name} - MAE: {mean\_absolute\_error(y\_test, y\_pre):.4f}',"\n")**

**a+=1**

**r2=r2\_score(y\_test, y\_pre)**

**if r2 > best\_r2:**

**best\_r2 = r2**

**best\_model\_name = name**

**best\_model = model**

**print(f'The best model is: {best\_model\_name}')**

**print(f'The best model R2 Score is: {best\_r2}')**

**data\_to\_predict = [[7420, 4, 2, 3, 1, 0, 0, 0, 1, 2, 1, 0]]**

**if best\_model is not None:**

**prediction = best\_model.predict(data\_to\_predict)**

**print("Best Model:", best\_model\_name)**

**print("Predicted Price:", prediction)**

**else:**

**print("No best model found. Something went wrong.")**

**IMPLEMENTATION IS CODED BY SUBALAKSHMI.S**