import pandas as pd import numpy as np import matplotlib.pyplot as plt import seabornassnsfromsklearn.model_selectionimporttrain_test_split,GridSearchCV from sklearn.preprocessing import StandardScaler, PolynomialFeatures from sklearn.linear_model import LinearRegression, Ridge, Lasso from sklearn.tree import DecisionTreeRegressor from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor from xgboost import XGBRegressor from sklearn.svm import SVR from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score from sklearn.pipeline import make_pipelineprint("\n=== LoadingData===")#Loaddataset(replacewithyour dataset)

url = "https://raw.githubusercontent.com/ageron/handsonml2/ master/datasets/housing/housing.csv" data = pd.read_csv(url) print(f"\nData Shape: {data.shape}")

print("\nFirst5Rows:")print(data.head())#BasicEDA Visualizations

plt.figure(figsize=(15, 10)) # Distribution of house prices plt.subplot(2, 2, 1) sns.histplot(data['median_house_value'], kde=True, bins=30) plt.title('House Price Distribution')

#Correlationheatmap plt.subplot(2, 2, 2)

#Selectonlynumericcolumnsnumeric_data= data.select_dtypes(include=['number'])

#Computecorrelationmatrixcorr=numeric_data.corr()# Plot heatmap sns.heatmap(corr, annot=True, cmap='coolwarm', fmt=".1f")

plt.title('FeatureCorrelation')# Price vs. median income plt.subplot(2, 2, 3)

sns.scatterplot(x='median_income',y='median_house_value',data=data,alpha=0.3) plt.title('Price vs. Income') # Price by ocean proximity

plt.subplot(2,2,4)

sns.boxplot(x='ocean_proximity',y='median_house_value',data=data) plt.xticks(rotation=45) plt.title('Price by Location') plt.tight_layout() plt.show() print("\n=== Preprocessing Data ===")

#Handlemissingvaluesdata.fillna(data.select_dtypes(include='number').median(), inplace=True)

Feature engineering data['rooms_per_household'] = data['total_rooms']/data['households'] data['bedrooms_per_room'] = data['total_bedrooms']/data['total_rooms'] # Convert categorical to numericaldata=pd.get_dummies(data,columns=['ocean_proximity'])

#Selectfeaturesand target

X=data.drop('median_house_value',axis=1)y

=data['median_house_value'] # Train-test split

X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=42) # Feature scaling scaler

= StandardScaler()

X_train_scaled=scaler.fit_transform(X_train) X_test_scaled = scaler.transform(X_test) print("\n===TrainingModels===")models=

{

"LinearRegression":LinearRegression(), "Ridge Regression": Ridge(alpha=1.0), "Lasso Regression": Lasso(alpha=0.1),

"DecisionTree":DecisionTreeRegressor(max_depth=5),

"Random Forest": RandomForestRegressor(n_estimators=100, random_state=42), "GradientBoosting":GradientBoostingRegressor(n_estimators=100,random_state=42), "XGBoost": XGBRegressor(n_estimators=100, random_state=42),

"SVR":SVR(kernel='rbf')

}results={}forname,modelin models.items():

print(f"Training {name}...") model.fit(X_train_scaled[:1000], y_train[:1000]) y_pred

= model.predict(X_test_scaled) results[name]={

"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)

}

Display results results_df = pd.DataFrame(results).T print("\n=== Model Performance ===") print(results_df.sort_values(by='RMSE')) print("\n=== Optimizing Best Model

===")

Let's optimize Random Forest as it typically performs well fromsklearn.model_selectionimportRandomizedSearchCV# Smaller parameter grid or use RandomizedSearchCV param_dist = {

'n_estimators':[50,100,200],

'max_depth':[None,10,20],

'min_samples_split':[2,5,10]

} rf = RandomForestRegressor(random_state=42)

random_search=RandomizedSearchCV(rf,param_distributions=param_dist,n_iter=5, cv=2, scoring='neg_mean_squared_error', n_jobs=-1,verbose=1,random_state=42,error_score='raise')random_search.fit(X_train_scaled,y_train) best_model = random_search.best_estimator_

Evaluate optimized model y_pred = best_model.predict(X_test_scaled)print("\nOptimized Model Performance:") print(f"MAE:

{mean_absolute_error(y_test, y_pred):.2f}") print(f"RMSE: {np.sqrt(mean_squared_error(y_test, y_pred)):.2f}") print(f"R2 Score: {r2_score(y_test,

y_pred):.4f}")print("\n===GeneratingVisualizations

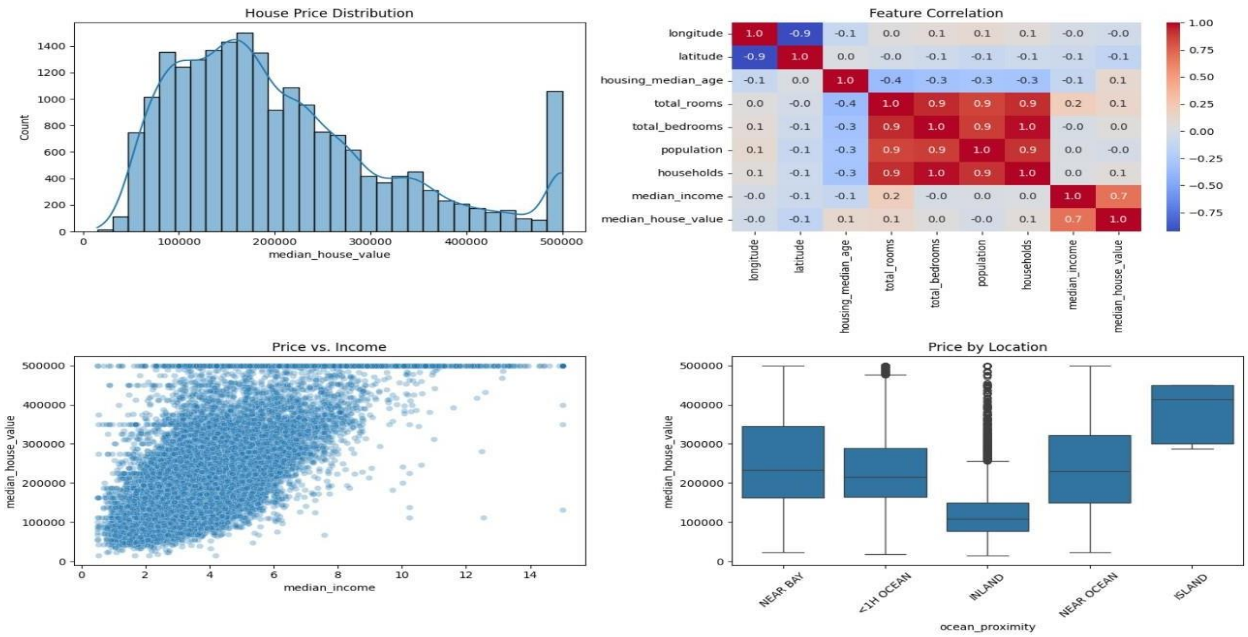
===")

#FeatureImportanceplt.figure(figsize=(10, 6))

importances = best_model.feature_importances_ features = X.columns indices = np.argsort(importances)[-10:]# Top 10 features plt.title('Feature Importances') plt.barh(range(len(indices)), importances[indices], color='b', align='center') plt.yticks(range(len(indices)), [features[i] for i in indices]) plt.xlabel('Relative Importance') plt.show() # Actual vs Predicted plt.figure(figsize=(10, 6)) plt.scatter(y_test, y_pred, alpha=0.3) plt.plot([y_test.min(),y_test.max()],[y_test.min(),y_test.max()],'k--',lw=2) plt.xlabel('Actual Prices') plt.ylabel('Predicted Prices') plt.title('Actual vs Predicted House Prices') plt.show() # Residual Plot residuals = y_test - y_pred plt.figure(figsize=(10, 6)) plt.scatter(y_pred, residuals, alpha=0.3) plt.axhline(y=0, color='r', linestyle='--') plt.xlabel('Predicted Prices') plt.ylabel('Residuals') plt.title('Residual Plot')

plt.show()print("\n===ProgramExecution Complete ===")

OUTPUT:


House Price Distribution
Count
median_house_value
Feature Correlation
longitude
latitude
housing_median_age
total_rooms
total_bedrooms
population
households
median_income
median_house_value
Price vs. Income
median_house_value
median_income
Price by Location
median_house_value
NEAR BAY
<1H OCEAN
INLAND
NEAR OCEAN
ISLAND
ocean_proximity