```
"""
House Price Forecasting Using Smart Regression Techniques
"""
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_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_pipeline

# =====================
# 1. Data Loading & EDA
# =====================
print("\n=== Loading Data ===")
# Load dataset (replace with your dataset)
url = "https://raw.githubusercontent.com/ageron/handson-ml2/master/datasets/housing/housin
data = pd.read_csv(url)

print(f"\nData Shape: {data.shape}")
print("\nFirst 5 Rows:")
print(data.head())

# Basic EDA 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')

# Correlation heatmap
plt.subplot(2, 2, 2)

# Select only numeric columns
numeric_data = data.select_dtypes(include=['number'])

# Compute correlation matrix
corr = numeric_data.corr()

# Plot heatmap
sns.heatmap(corr, annot=True, cmap='coolwarm', fmt=".1f")
plt.title('Feature Correlation')
```

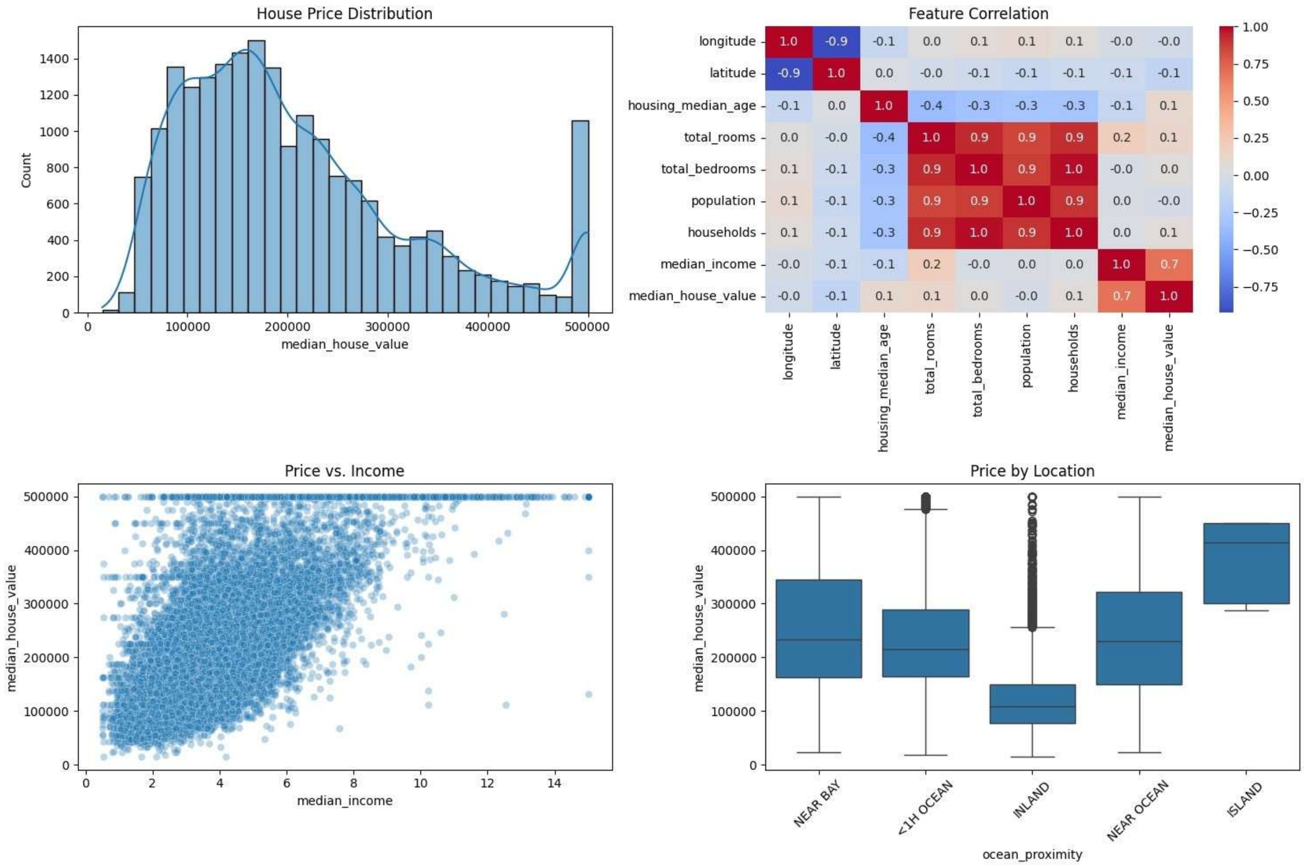
```
# Price vs. median income
plt.subplot(2, 2, 3)
sns.scatterplot(x='median_income', y='median_house_value', data=data, alpha=0.3)
plt.title('Price vs. Income')

# 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()

# =====================
# 2. Data Preprocessing
# =====================
print("\n=== Preprocessing Data ===")
# Handle missing values
data.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 numerical
data = pd.get_dummies(data, columns=['ocean_proximity'])

# Select features and 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)

# =====================
# 3. Model Training
# =====================
print("\n=== Training Models ===")
models = {
    "Linear Regression": LinearRegression(),
    "Ridge Regression": Ridge(alpha=1.0),
    "Lasso Regression": Lasso(alpha=0.1),
    "Decision Tree": DecisionTreeRegressor(max_depth=5),
    "Random Forest": RandomForestRegressor(n_estimators=100, random_state=42),
     Gradient Boosting   GradientBoostingRegressor  _estimators= 0   random_state=4
```

```
    "XGBoost": XGBRegressor(n_estimators=100, random_state=42),
    "SVR": SVR(kernel='rbf')
}

results = {}
for name, model in 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'))

# =====================
# 4. Model Optimization
# =====================
print("\n=== Optimizing Best Model ===")
# Let's optimize Random Forest as it typically performs well
from sklearn.model_selection import RandomizedSearchCV

# 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,  sc
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}")

# =====================
# 5. Visualization
```

```
# =====================
print("\n=== Generating Visualizations ===")
# Feature Importance
plt.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=== Program Execution Complete ===")
```

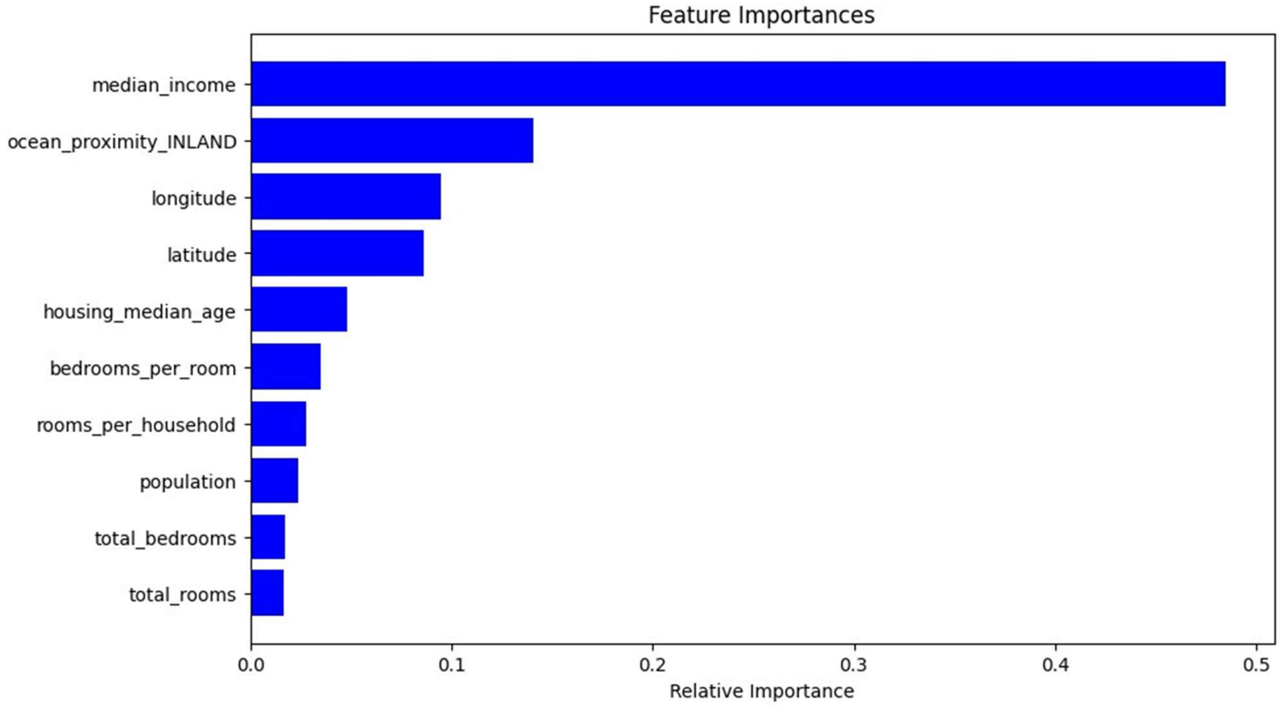
```
=== Loading Data ===

Data Shape: (20640, 10)

First 5 Rows:
   longitude  latitude  housing_median_age  total_rooms  total_bedrooms  \
0    -122.23     37.88                41.0        880.0           129.0
1    -122.22     37.86                21.0       7099.0          1106.0
2    -122.24     37.85                52.0       1467.0           190.0
3    -122.25     37.85                52.0       1274.0           235.0
4    -122.25     37.85                52.0       1627.0           280.0

   population  households  median_income  median_house_value ocean_proximity
0       322.0       126.0         8.3252            452600.0        NEAR BAY
1      2401.0      1138.0         8.3014            358500.0        NEAR BAY
2       496.0       177.0         7.2574            352100.0        NEAR BAY
3       558.0       219.0         5.6431            341300.0        NEAR BAY
```

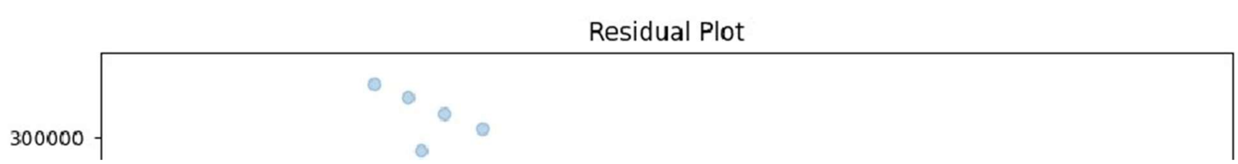
```
4        565.0        259.0        3.8462        342200.0        NEAR BAY
```

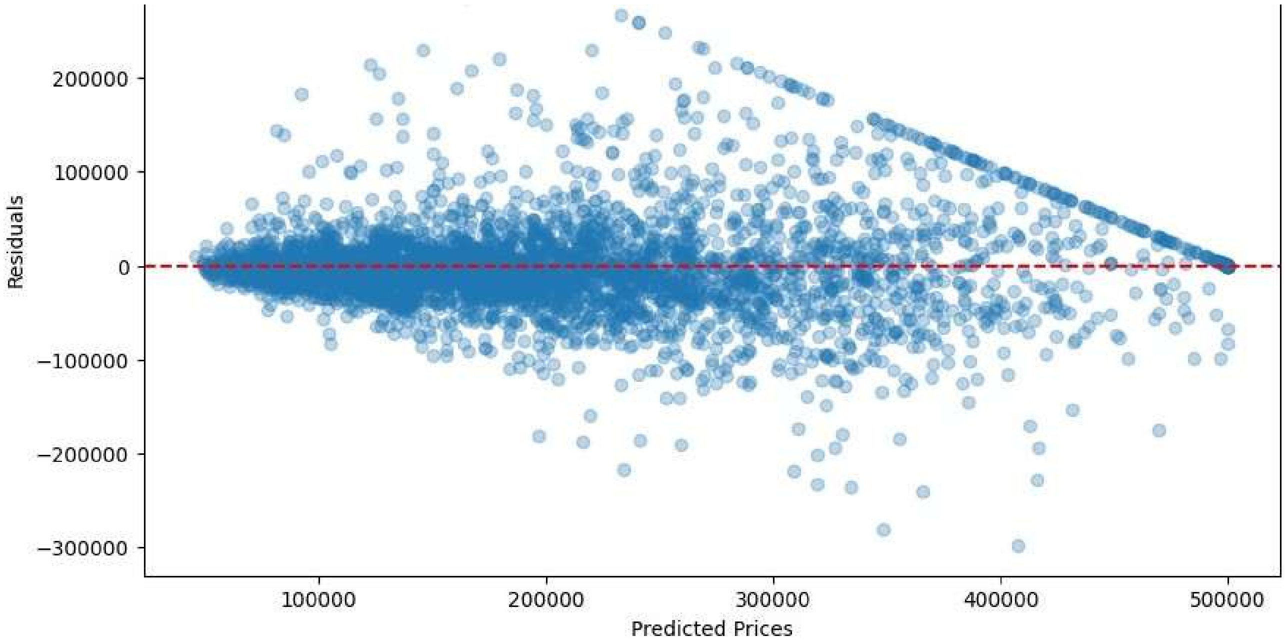



```
=== Preprocessing Data ===

=== Training Models ===
Training Linear Regression...
Training Ridge Regression...
Training Lasso Regression...
Training Decision Tree...
/usr/local/lib/python3.11/dist-packages/sklearn/linear_model/_coordinate_descent.py:6
  model = cd_fast.enet_coordinate_descent(
Training Random Forest...
Training Gradient Boosting...
Training XGBoost...
Training SVR...

=== Model Performance ===
                            MAE           RMSE        R2
Gradient Boosting   42899.681862   61706.179667  0.709430
XGBoost             43078.089212   62689.820720  0.700093
Random Forest       44795.868522   64658.348085  0.680962
Ridge Regression    50698.989573   70888.282593  0.616521
Linear Regression   50780.097069   71026.417101  0.615025
Lasso Regression    50780.319784   71027.442353  0.615014
Decision Tree       52042.762451   75685.545444  0.562862
SVR                 87488.892642  116372.585393 -0.033462

=== Optimizing Best Model ===
Fitting 2 folds for each of 5 candidates, totalling 10 fits
```

```
Optimized Model Performance:
MAE: 32525.65
RMSE: 50034.55
R2 Score: 0.8090

=== Generating Visualizations ===
```








```
=== Program Execution Complete ===
```