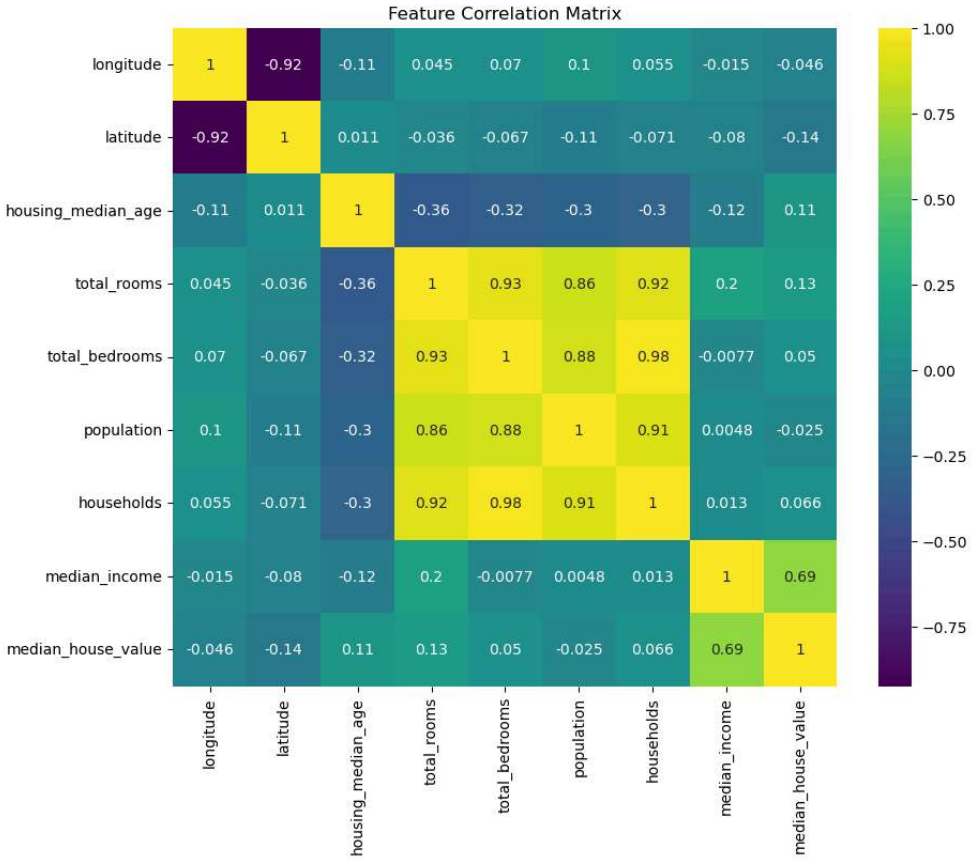
In [15]:
```
import pandas as pd


df = pd.read_csv(r'C:\Users\humer\Downloads\california.housing.csv


print(df.head())
print(df.info())
```

```
   longitude  latitude  housing_median_age  total_rooms  total_bedr
ooms  \
0    -122.23     37.88                  41          880           1
29.0
1    -122.22     37.86                  21         7099          11
06.0
2    -122.24     37.85                  52         1467           1
90.0
3    -122.25     37.85                  52         1274           2
35.0
4    -122.25     37.85                  52         1627           2
80.0

   population  households  median_income  median_house_value ocean_
proximity
0         322         126         8.3252              452600
NEAR BAY
1        2401        1138         8.3014              358500
NEAR BAY
2         496         177         7.2574              352100
NEAR BAY
3         558         219         5.6431              341300
NEAR BAY
4         565         259         3.8462              342200
NEAR BAY
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20640 entries, 0 to 20639
Data columns (total 10 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   longitude           20640 non-null  float64
 1   latitude            20640 non-null  float64
 2   housing_median_age  20640 non-null  int64
 3   total_rooms         20640 non-null  int64
 4   total_bedrooms      20433 non-null  float64
 5   population          20640 non-null  int64
 6   households          20640 non-null  int64
 7   median_income       20640 non-null  float64
 8   median_house_value  20640 non-null  int64
 9   ocean_proximity     20640 non-null  object
dtypes: float64(4), int64(5), object(1)
memory usage: 1.6+ MB
None
```

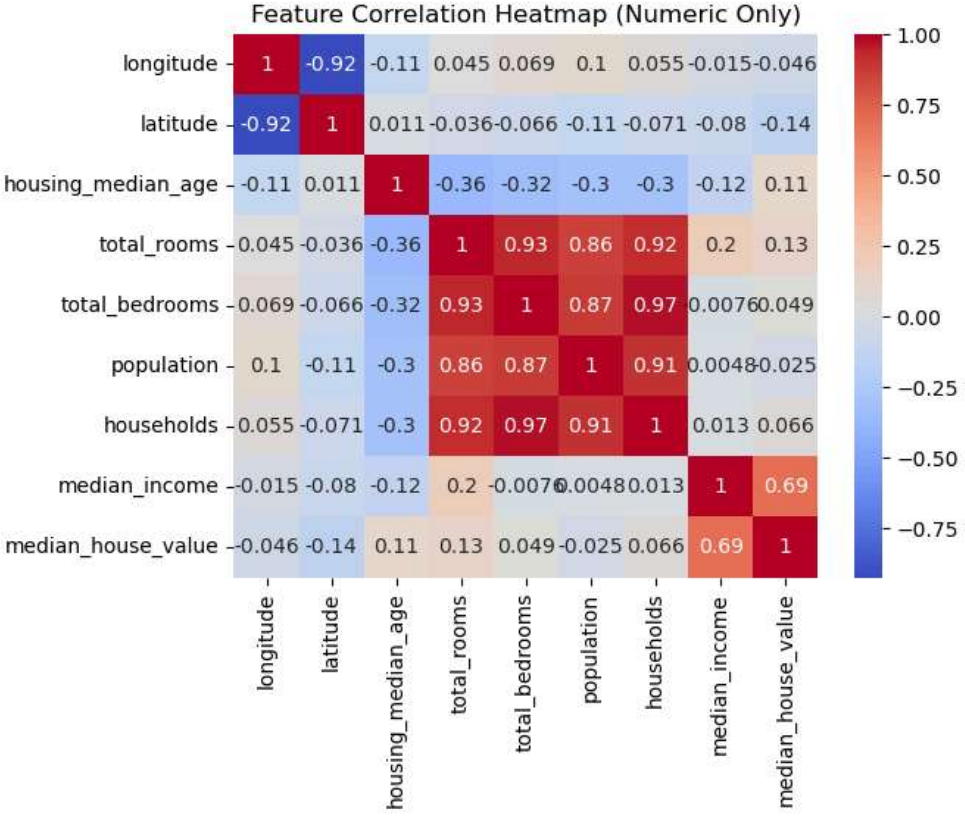
In [16]:
```
numeric_df = df.select_dtypes(include=['float64', 'int64'])

plt.figure(figsize=(10, 8))
sns.heatmap(numeric_df.corr(), annot=True, cmap='viridis')
plt.title("Feature Correlation Matrix")
plt.show()
```



In [40]:
```
print(df.info())
print(df.describe())


print(df.isnull().sum())

# Correlation heatmap
import seaborn as sns
import matplotlib.pyplot as plt
df_numeric = df.drop(columns=['ocean_proximity'])
sns.heatmap(df_numeric.corr(), annot=True, cmap='coolwarm')
plt.title("Feature Correlation Heatmap (Numeric Only)")
plt.show()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20640 entries, 0 to 20639
Data columns (total 10 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   longitude           20640 non-null  float64
 1   latitude            20640 non-null  float64
 2   housing_median_age  20640 non-null  int64  
 3   total_rooms         20640 non-null  int64  
 4   total_bedrooms      20640 non-null  float64
 5   population          20640 non-null  int64  
 6   households          20640 non-null  int64  
 7   median_income       20640 non-null  float64
 8   median_house_value  20640 non-null  int64  
 9   ocean_proximity     20640 non-null  object 
dtypes: float64(4), int64(5), object(1)
memory usage: 1.6+ MB
None
          longitude      latitude  housing_median_age   total_rooms
\
count  20640.000000  20640.000000        20640.000000  20640.000000
mean    -119.569704     35.631861           28.639486   2635.763081
std        2.003532      2.135952           12.585558   2181.615252
min     -124.350000     32.540000            1.000000      2.000000
25%     -121.800000     33.930000           18.000000   1447.750000
50%     -118.490000     34.260000           29.000000   2127.000000
75%     -118.010000     37.710000           37.000000   3148.000000
max     -114.310000     41.950000           52.000000  39320.000000

       total_bedrooms    population    households  median_income  \
count    20640.000000  20640.000000  20640.000000   20640.000000
mean       536.838857   1425.476744    499.539680       3.870671
std        419.391878   1132.462122    382.329753       1.899822
min          1.000000      3.000000      1.000000       0.499900
25%        297.000000    787.000000    280.000000       2.563400
50%        435.000000   1166.000000    409.000000       3.534800
75%        643.250000   1725.000000    605.000000       4.743250
max       6445.000000  35682.000000   6082.000000      15.000100

       median_house_value
count        20640.000000
mean        206855.816909
std         115395.615874
min          14999.000000
25%         119600.000000
50%         179700.000000
75%         264725.000000
max         500001.000000
longitude             0
latitude              0
housing_median_age    0
total_rooms           0
total_bedrooms        0
```

```
population          0
households          0
median_income       0
median_house_value  0
ocean_proximity     0
dtype: int64
```



In [25]:
```
from sklearn.linear_model import LinearRegression


lr_model = LinearRegression()
lr_model.fit(X_train, y_train)
```

Out[25]:
```
LinearRegression()
```

In [35]:
```
# Predict using the trained model
y_pred_lr = lr_model.predict(X_test)


print("Linear Regression Performance:")
print("R² Score:", r2_score(y_test, y_pred_lr))
print("MAE:", mean_absolute_error(y_test, y_pred_lr))
import numpy as np
```

```
rmse = np.sqrt(mean_squared_error(y_test, y_pred_lr))
print("RMSE:", rmse)
```

```
Linear Regression Performance:
R² Score: 0.6254240620553608
MAE: 50670.7382409719
RMSE: 70060.52184473517
```

In [34]:
```
from sklearn.ensemble import RandomForestRegressor

rf_model = RandomForestRegressor(random_state=42)
rf_model.fit(X_train, y_train)
y_pred_rf = rf_model.predict(X_test)

print("Random Forest Performance:")
print("R² Score:", r2_score(y_test, y_pred_rf))
print("MAE:", mean_absolute_error(y_test, y_pred_rf))
import numpy as np

rmse_rf = np.sqrt(mean_squared_error(y_test, y_pred_rf))
print("RMSE:", rmse_rf)
```

```
Random Forest Performance:
R² Score: 0.8169555593071559
MAE: 31643.65566860465
RMSE: 48975.818369986104
```

In [47]:
```
print(X.dtypes)
```

```
longitude             float64
latitude              float64
housing_median_age      int64
total_rooms             int64
total_bedrooms        float64
population              int64
households              int64
median_income         float64
ocean_proximity        object
dtype: object
```

In [ ]: