Assignment No: 3

```
import pandas as pd
df = pd.read_csv("/content/sample_data/Housing.csv")  # Replace with your actual dataset file
print(df.head())
```

Output:

```
      price  area  bedrooms  bathrooms  stories mainroad guestroom basement  \
0  13300000  7420         4          2        3      yes        no       no
1  12250000  8960         4          4        4      yes        no       no
2  12250000  9960         3          2        2      yes        no      yes
3  12215000  7500         4          2        2      yes        no      yes
4  11410000  7420         4          1        2      yes       yes      yes

  hotwaterheating airconditioning  parking prefarea furnishingstatus
0              no             yes        2      yes        furnished
1              no             yes        3       no        furnished
2              no              no        2      yes   semi-furnished
3              no             yes        3      yes        furnished
4              no             yes        2       no        furnished
```

```
import pandas as pd
df = pd.read_csv("/content/sample_data/Housing.csv")
print(df.dtypes)
```

Output:

```
price              int64
area               int64
bedrooms           int64
bathrooms          int64
stories            int64
mainroad          object
guestroom         object
basement          object
```

```
hotwaterheating     object
airconditioning     object
parking              int64
prefarea            object
furnishingstatus    object
dtype: object
```

```
import pandas as pd
df = pd.read_csv("/content/sample_data/Housing.csv
# Count column types
num_categorical = df.select_dtypes(include=['object']).shape[1]
num_integer = df.select_dtypes(include=['int64']).shape[1]
num_float = df.select_dtypes(include=['float64']).shape[1]

# Display results
print(f"Number of Categorical Columns: {num_categorical}")
print(f"Number of Integer Columns: {num_integer}")
print(f"Number of Float Columns: {num_float}")
```

Output: Number of Categorical Columns: 7

```
Number of Integer Columns: 6
Number of Float Columns: 0
```

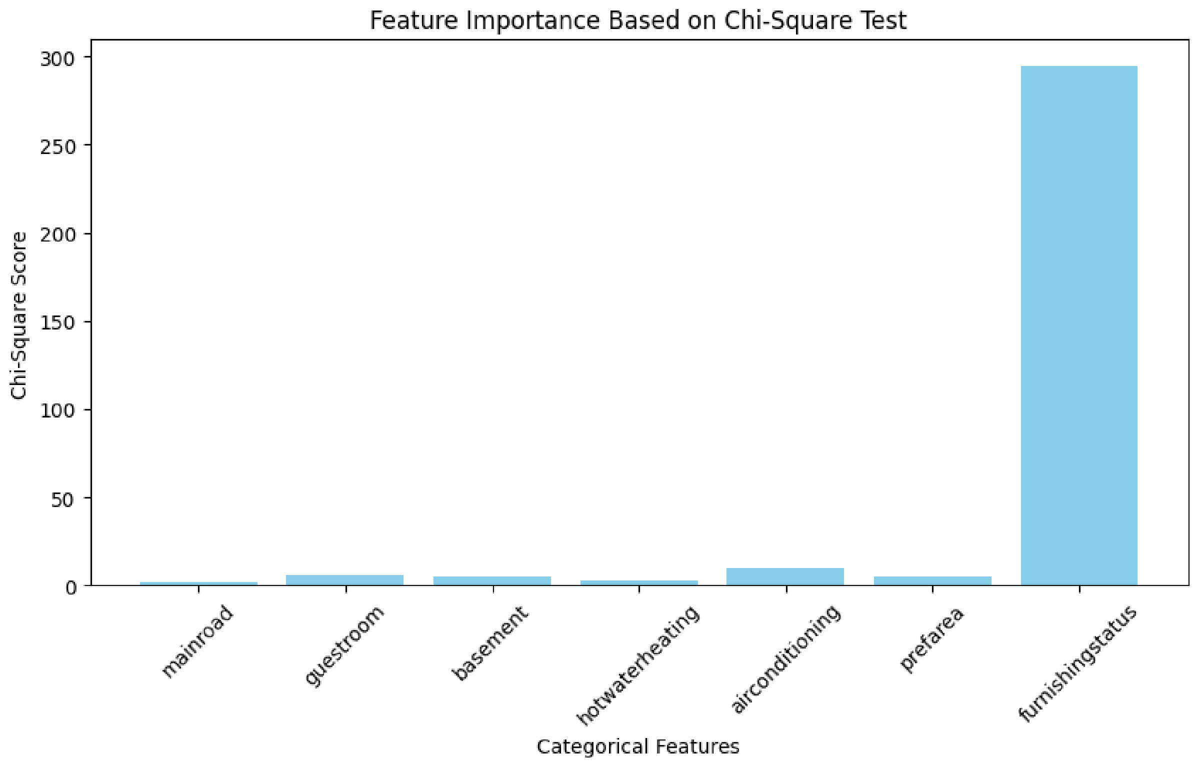
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.feature_selection import chi2
from sklearn.preprocessing import LabelEncoder

df = pd.read_csv("/content/sample_data/Housing.csv")
categorical_cols = df.select_dtypes(include=['object']).columns
```

```
encoder = LabelEncoder()
df_encoded = df.copy()
for col in categorical_cols:
    df_encoded[col] = encoder.fit_transform(df[col])

X = df_encoded[categorical_cols]  # Features
y = df_encoded.iloc[:, -1]  # Target variable

chi_scores, p_values = chi2(X, y)
plt.figure(figsize=(10, 5))
plt.bar(categorical_cols, chi_scores, color='skyblue')
plt.xlabel("Categorical Features")
plt.ylabel("Chi-Square Score")
plt.title("Feature Importance Based on Chi-Square Test")
plt.xticks(rotation=45)
plt.show()
```



```
new_df=df.dropna()
new_df.isnull().sum()
```

Output: 0

```
price	0
area	0
bedrooms	0
bathrooms	0
stories	0
mainroad	0
guestroom	0
basement	0
hotwaterheating	0
airconditioning	0
parking	0
prefarea	0
furnishingstatus	0

dtype: int64
```

```
import pandas as pd
categorical_cols = df.select_dtypes(include=['object']).columns
df[categorical_cols] = df[categorical_cols].replace({'Yes': 1, 'No': 0})

df_encoded = pd.get_dummies(df, columns=categorical_cols, drop_first=True)
print(df_encoded.head().to_string(index=False))

import pandas as pd
from sklearn.model_selection import train_test_split
X = df.drop(columns=['price'])  # Features (all columns except 'Price')
y = df['price']  # Target variable
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
print("Training Features Shape:", X_train.shape)
print("Testing Features Shape:", X_test.shape)
print("Training Target Shape:", y_train.shape)
print("Testing Target Shape:", y_test.shape)
```

Output: Training Features Shape: (436, 12)

Testing Features Shape: (109, 12)

Training Target Shape: (436,)

Testing Target Shape: (109,)

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVR
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
# Preprocess data
X = pd.get_dummies(df.drop(columns=['price']), drop_first=True)  # One-Hot Encoding
y = df['price']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Standardize features
scaler = StandardScaler()
X_train, X_test = scaler.fit_transform(X_train), scaler.transform(X_test)

# Train & predict with SVM
svm = SVR(kernel='rbf', C=100, gamma=0.1, epsilon=0.1)
svm.fit(X_train, y_train)
y_pred = svm.predict(X_test)

# Evaluate model
```

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

Output: MAE: 1762754.37, RMSE: 2358789.58, R^2: -0.10

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

X = pd.get_dummies(df.drop(columns=['price']), drop_first=True)  # One-Hot Encoding for categorical data
y = df['price']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Standardize features (optional for tree-based models, but good practice)
scaler = StandardScaler()
X_train, X_test = scaler.fit_transform(X_train), scaler.transform(X_test)

# Train & predict with Random Forest
rf = RandomForestRegressor(n_estimators=100, random_state=42)
rf.fit(X_train, y_train)
y_pred = rf.predict(X_test)

# Evaluate model
print(f"MAE: {mean_absolute_error(y_test, y_pred):.2f}, RMSE: {np.sqrt(mean_squared_error(y_test, y_pred)):.2f}, R²: {r2_score(y_test, y_pred):.2f}")
```

Output: MAE: 1021151.08, RMSE: 1399758.20, R^2: 0.61

```
from sklearn.metrics import r2_score
```

```
rf = RandomForestRegressor(n_estimators=100, random_state=42)
rf.fit(X_train, y_train)
r2 = r2_score(y_test, rf.predict(X_test))
print(f"R² Score: {r2:.2f}")
```

Output: R^2 Score: 0.61

```
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
lr.fit(X_train, y_train)
r2 = r2_score(y_test, lr.predict(X_test))
print(f"R² Score: {r2:.2f}")
```

Output: R^2 Score: 0.65

```
from sklearn.linear_model import Lasso

lasso = Lasso(alpha=0.1)  # Adjust alpha for regularization strength
lasso.fit(X_train, y_train)
r2 = r2_score(y_test, lasso.predict(X_test))
print(f"R² Score: {r2:.2f}")
```

R^2 Score: 0.65