BDA - ASSIGNMENT

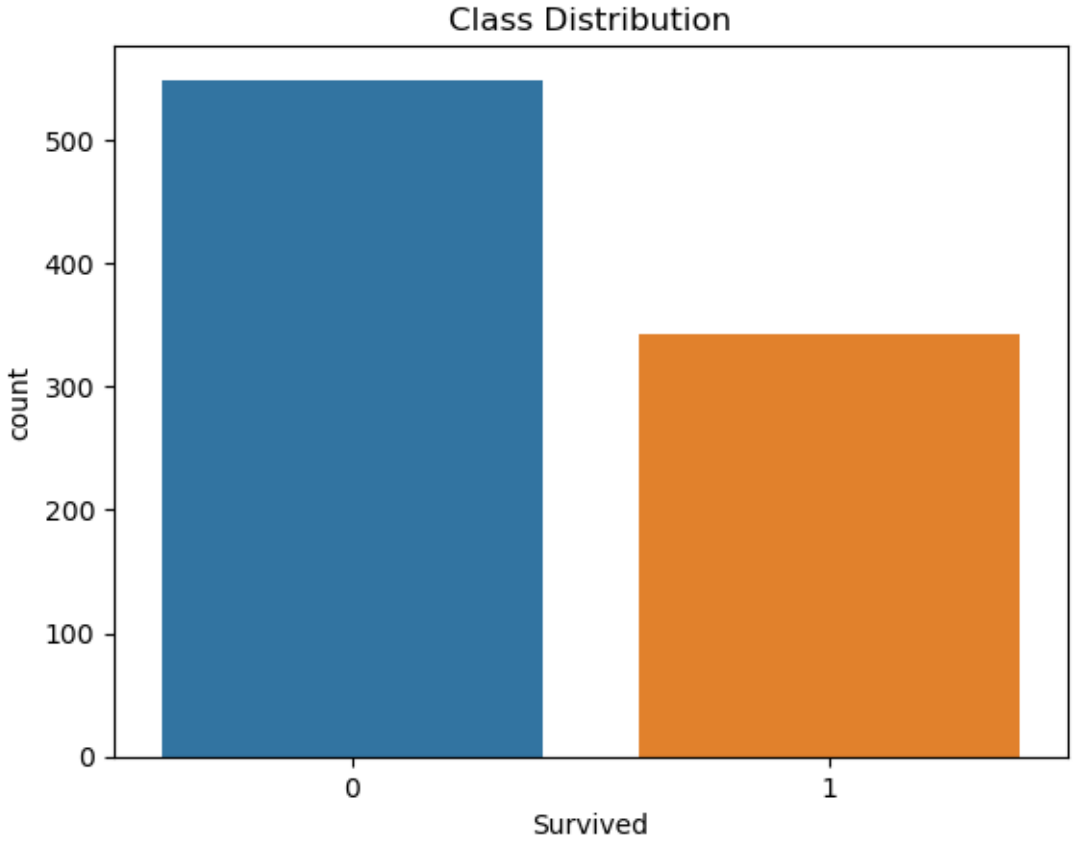
1 1. How can you use Python to handle imbalanced datasets for classification tasks?

[1]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix
from imblearn.over_sampling import SMOTE  # Correct import statement
import matplotlib.pyplot as plt
import seaborn as sns

# Load Titanic dataset
df = pd.read_csv('titanic.csv')

# Display the first few rows and the class distribution
print(df.head())
print("\nClass Distribution:\n", df['Survived'].value_counts())

# Plot class distribution
sns.countplot(x='Survived', data=df)
plt.title("Class Distribution")
plt.show()

# Preprocess the dataset
# Fill missing values
df['Age'].fillna(df['Age'].median(), inplace=True)
df['Embarked'].fillna(df['Embarked'].mode()[0], inplace=True)

# Convert categorical columns to numerical using one-hot encoding
df = pd.get_dummies(df, columns=['Sex', 'Embarked'], drop_first=True)

# Drop unnecessary columns
df.drop(['PassengerId', 'Name', 'Ticket', 'Cabin'], axis=1, inplace=True)

# Separate features and target variable
X = df.drop('Survived', axis=1)
```

```
y = df['Survived']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
 ↪random_state=42)

# Display class distribution before SMOTE
print("\nTraining class distribution:", np.bincount(y_train))
print("Testing class distribution:", np.bincount(y_test))

# Apply SMOTE to handle class imbalance
smote = SMOTE(random_state=42)
X_train_resampled, y_train_resampled = smote.fit_resample(X_train, y_train)

# Display class distribution after SMOTE
print("\nResampled training class distribution:", np.
 ↪bincount(y_train_resampled))

# Train a Random Forest Classifier
clf = RandomForestClassifier(random_state=42)
clf.fit(X_train_resampled, y_train_resampled)

# Predict on the test set
y_pred = clf.predict(X_test)

# Evaluate the model
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))

# Plot confusion matrix heatmap
plt.figure(figsize=(8, 4))
sns.heatmap(confusion_matrix(y_test, y_pred), annot=True, fmt="d", cmap="Blues")
plt.title("Confusion Matrix Heatmap")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
```

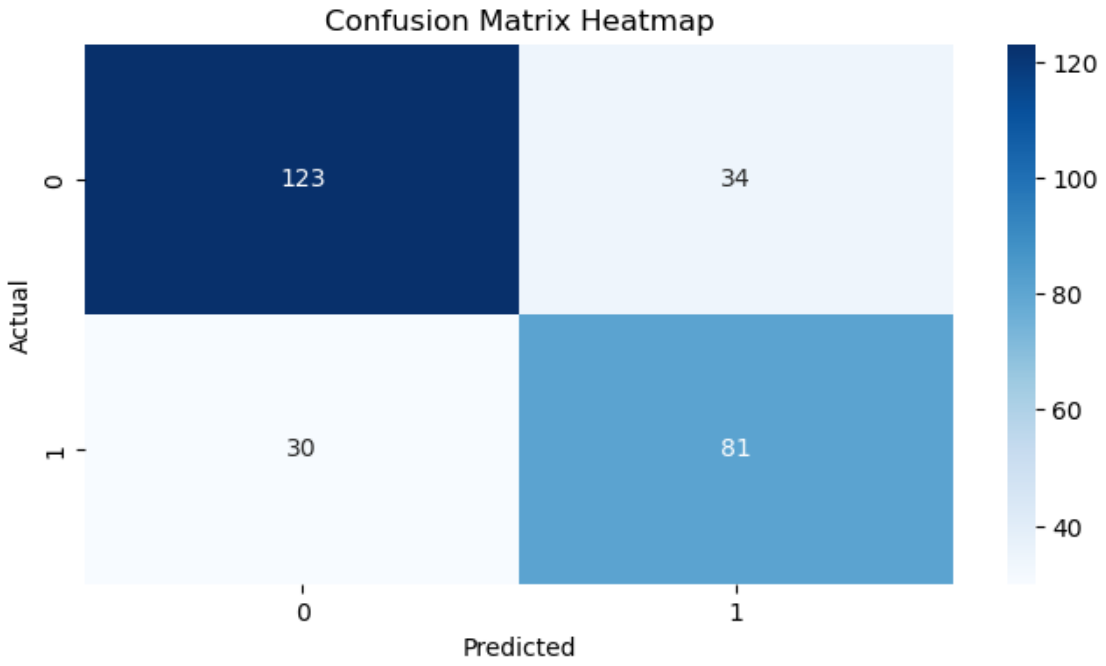
```
   PassengerId  Survived  Pclass  \
0            1         0       3
1            2         1       1
2            3         1       3
3            4         1       1
4            5         0       3

                                                Name     Sex   Age  SibSp  \
0                            Braund, Mr. Owen Harris    male  22.0      1
1  Cumings, Mrs. John Bradley (Florence Briggs Th…  female  38.0      1
```

```
2                             Heikkinen, Miss. Laina  female  26.0      0
3       Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
4                           Allen, Mr. William Henry    male  35.0      0

   Parch            Ticket     Fare Cabin Embarked
0      0         A/5 21171   7.2500   NaN        S
1      0          PC 17599  71.2833   C85        C
2      0  STON/O2. 3101282   7.9250   NaN        S
3      0            113803  53.1000  C123        S
4      0            373450   8.0500   NaN        S

Class Distribution:
 0    549
1    342
Name: Survived, dtype: int64
```




```
Training class distribution: [392 231]
Testing class distribution: [157 111]

Resampled training class distribution: [392 392]
```

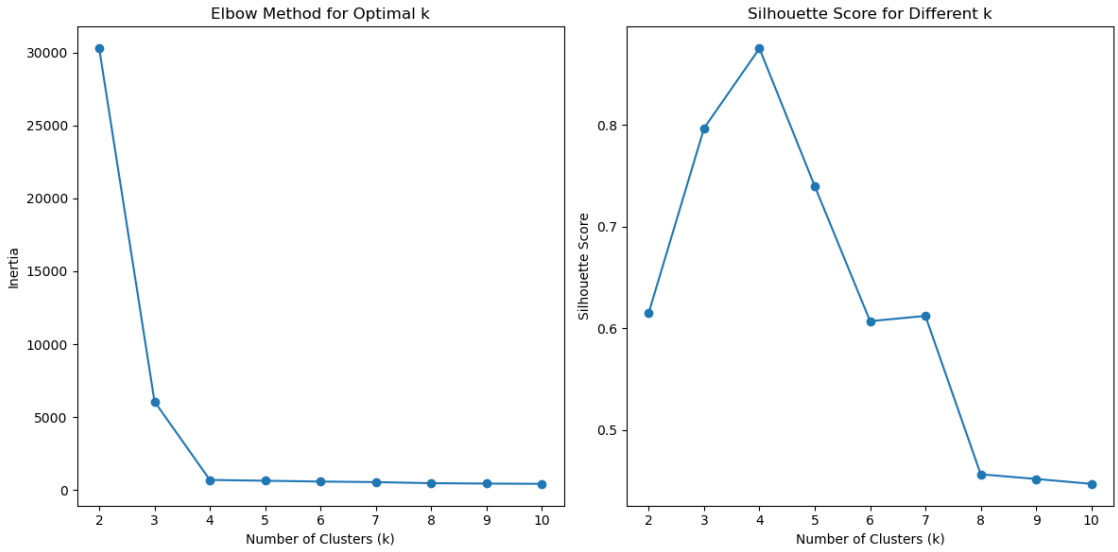
```
Confusion Matrix:
 [[123  34]
 [ 30  81]]

Classification Report:
              precision    recall  f1-score   support

           0       0.80      0.78      0.79       157
           1       0.70      0.73      0.72       111

    accuracy                           0.76       268
   macro avg       0.75      0.76      0.76       268
weighted avg       0.76      0.76      0.76       268
```



2 2. How do you choose the optimal number of clusters for K-means in Python?

[2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.datasets import make_blobs
```

```
from sklearn.metrics import silhouette_score

# Generate a synthetic dataset
X, _ = make_blobs(n_samples=1000, centers=4, cluster_std=0.60, random_state=42)

# Elbow Method to find the optimal number of clusters
inertia = []
silhouette_scores = []

# Try different values of k (from 2 to 10 clusters)
for k in range(2, 11):
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(X)

    # Calculate inertia (within-cluster sum of squares)
    inertia.append(kmeans.inertia_)

    # Calculate silhouette score
    silhouette_avg = silhouette_score(X, kmeans.labels_)
    silhouette_scores.append(silhouette_avg)

# Plot Elbow Method (Inertia vs k)
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
plt.plot(range(2, 11), inertia, marker='o')
plt.title("Elbow Method for Optimal k")
plt.xlabel("Number of Clusters (k)")
plt.ylabel("Inertia")

# Plot Silhouette Scores
plt.subplot(1, 2, 2)
plt.plot(range(2, 11), silhouette_scores, marker='o')
plt.title("Silhouette Score for Different k")
plt.xlabel("Number of Clusters (k)")
plt.ylabel("Silhouette Score")

plt.tight_layout()
plt.show()

# From the plots, choose the optimal k (usually the elbow point for inertia and␣
 ↪highest silhouette score)
```

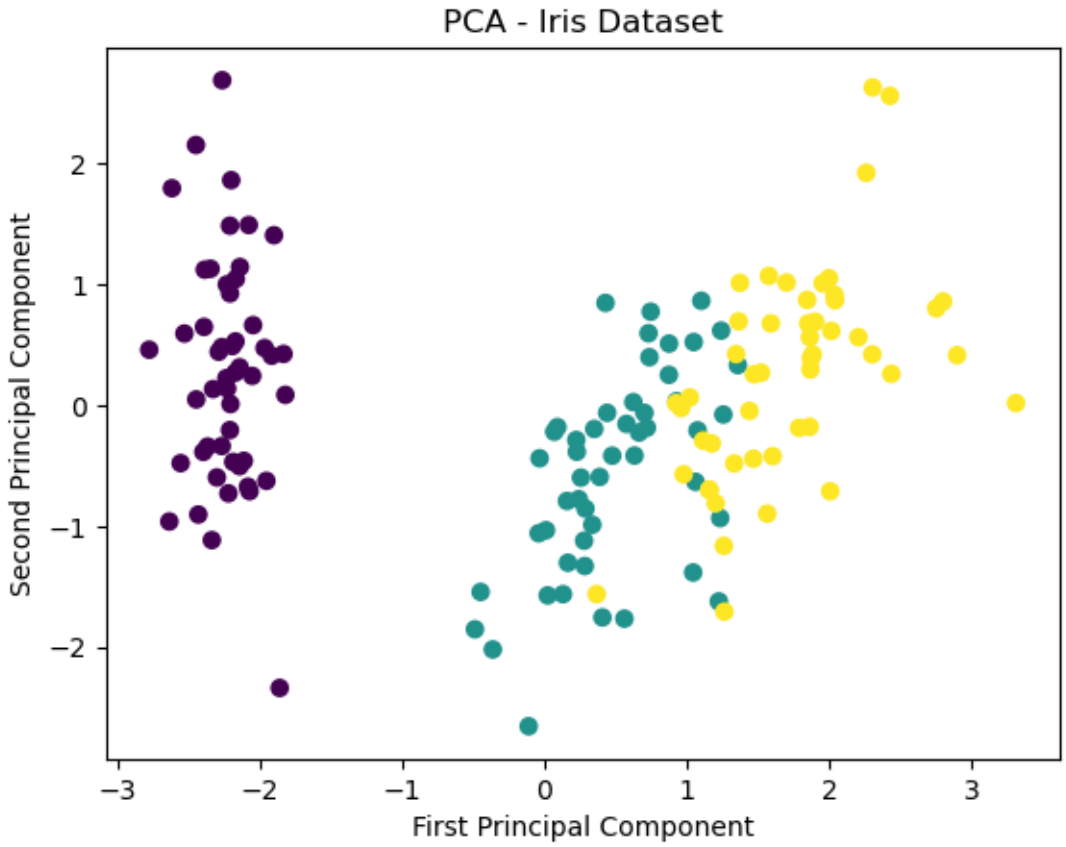


3 # 3. What techniques can you use to reduce dimensionality for large datasets (e.g., PCA)?

[3]:
```
import pandas as pd
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt

# Load dataset (example using the Iris dataset)
from sklearn.datasets import load_iris
data = load_iris()
X = data.data

# Standardize the data
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Apply PCA
pca = PCA(n_components=2)  # Reduce to 2 components
X_pca = pca.fit_transform(X_scaled)

# Explained Variance
print(f"Explained variance ratio: {pca.explained_variance_ratio_}")

# Plot the first two principal components
plt.scatter(X_pca[:, 0], X_pca[:, 1], c=data.target)
plt.xlabel('First Principal Component')
```

```
plt.ylabel('Second Principal Component')
plt.title('PCA - Iris Dataset')
plt.show()
```

```
Explained variance ratio: [0.72962445 0.22850762]
```



4 4. How do you use Python to find and visualize correlations in a big dataset?

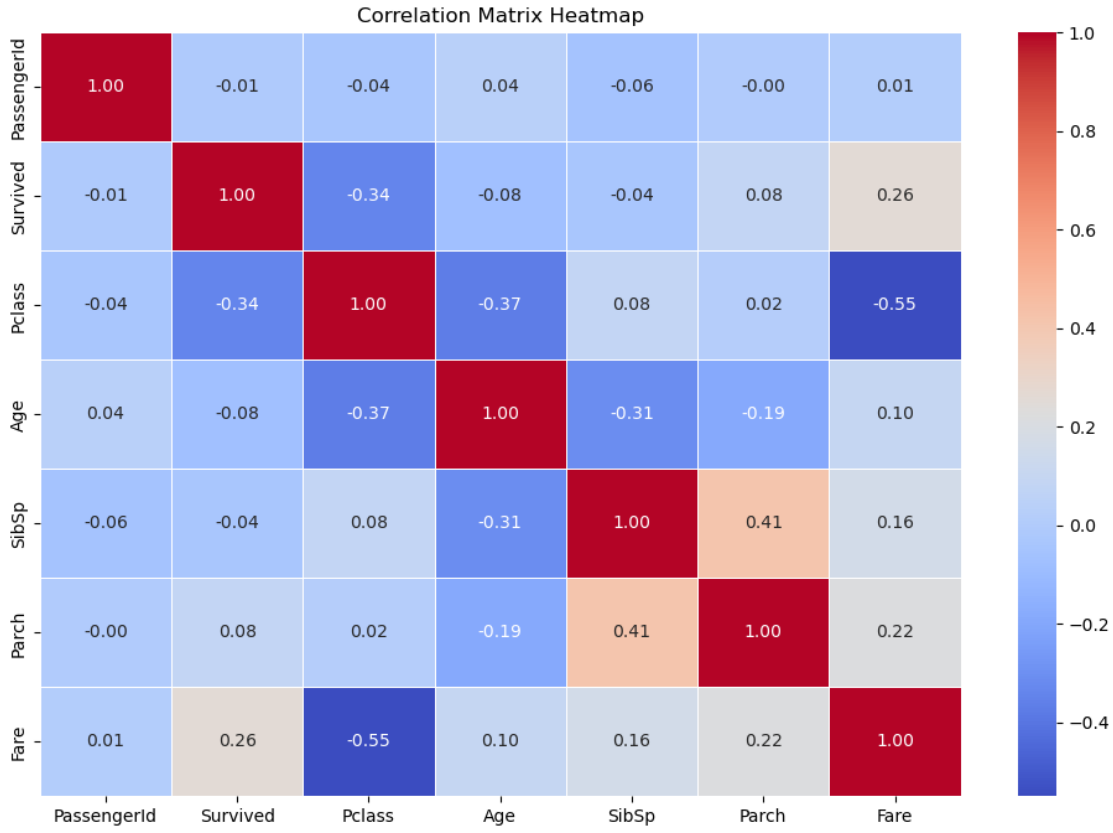
[4]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Load a sample dataset (e.g., the Titanic dataset)
df = pd.read_csv('titanic.csv')

# Calculate the correlation matrix
correlation_matrix = df.corr()
```

```
# Display the correlation matrix
print(correlation_matrix)

# Visualize the correlation matrix with a heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f',␣
 ↪linewidths=0.5)
plt.title('Correlation Matrix Heatmap')
plt.show()
```

```
C:\Users\aishwarya\AppData\Local\Temp\ipykernel_21332\3076006076.py:9:
FutureWarning: The default value of numeric_only in DataFrame.corr is
deprecated. In a future version, it will default to False. Select only valid
columns or specify the value of numeric_only to silence this warning.
  correlation_matrix = df.corr()
```

	PassengerId	Survived	Pclass	Age	SibSp	Parch	\
PassengerId	1.000000	-0.005007	-0.035144	0.036847	-0.057527	-0.001652	
Survived	-0.005007	1.000000	-0.338481	-0.077221	-0.035322	0.081629	
Pclass	-0.035144	-0.338481	1.000000	-0.369226	0.083081	0.018443	
Age	0.036847	-0.077221	-0.369226	1.000000	-0.308247	-0.189119	
SibSp	-0.057527	-0.035322	0.083081	-0.308247	1.000000	0.414838	
Parch	-0.001652	0.081629	0.018443	-0.189119	0.414838	1.000000	
Fare	0.012658	0.257307	-0.549500	0.096067	0.159651	0.216225	

	Fare
PassengerId	0.012658
Survived	0.257307
Pclass	-0.549500
Age	0.096067
SibSp	0.159651
Parch	0.216225
Fare	1.000000



5 5. How can you handle missing values in a dataset using Python?

[5]:
```
import pandas as pd
import numpy as np

# Load Titanic dataset (assuming the dataset is already loaded in the variable␣
↪`df`)
df = pd.read_csv('titanic.csv')

# Display the original dataset
print("Original Dataset:")
print(df.head())

# Count missing values in each column
print("\nMissing Value Count:")
print(df.isnull().sum())

# 1. Drop rows with missing values
```

```
df_dropped = df.dropna()
print("\nDataset after Dropping Rows with Missing Values:")
print(df_dropped.head())

# 2. Fill missing values with mean for numerical columns
df_filled_mean = df.fillna({
    'Age': df['Age'].mean(),
    'Fare': df['Fare'].mean(),
    # Fill other numerical columns if needed, for example:
    # 'SibSp': df['SibSp'].mean(),
    # 'Parch': df['Parch'].mean()
})
print("\nDataset after Filling Missing Values with Mean for Numerical Columns:")
print(df_filled_mean.head())

# 3. Forward fill (use previous valid value)
df_ffill = df.fillna(method='ffill')
print("\nDataset after Forward Fill:")
print(df_ffill.head())

# 4. Backward fill (use next valid value)
df_bfill = df.fillna(method='bfill')
print("\nDataset after Backward Fill:")
print(df_bfill.head())

# 5. Interpolate missing values for numerical columns
df_interpolated = df.copy()
df_interpolated['Age'] = df_interpolated['Age'].interpolate(method='linear')
df_interpolated['Fare'] = df_interpolated['Fare'].interpolate(method='linear')
# Apply interpolation to other numerical columns as needed
print("\nDataset after Interpolation for Numerical Columns:")
print(df_interpolated.head())
```

```
Original Dataset:
   PassengerId  Survived  Pclass  \
0            1         0       3
1            2         1       1
2            3         1       3
3            4         1       1
4            5         0       3

                                                Name     Sex   Age  SibSp  \
0                            Braund, Mr. Owen Harris    male  22.0      1
1  Cumings, Mrs. John Bradley (Florence Briggs Th…  female  38.0      1
2                             Heikkinen, Miss. Laina  female  26.0      0
3       Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
4                           Allen, Mr. William Henry    male  35.0      0
```

```
   Parch            Ticket     Fare Cabin Embarked
0      0         A/5 21171   7.2500   NaN        S
1      0          PC 17599  71.2833   C85        C
2      0  STON/O2. 3101282   7.9250   NaN        S
3      0            113803  53.1000  C123        S
4      0            373450   8.0500   NaN        S

Missing Value Count:
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64

Dataset after Dropping Rows with Missing Values:
    PassengerId  Survived  Pclass  \
1             2         1       1
3             4         1       1
6             7         0       1
10           11         1       3
11           12         1       1

                                                 Name     Sex   Age  SibSp  \
1   Cumings, Mrs. John Bradley (Florence Briggs Th…  female  38.0      1
3        Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
6                             McCarthy, Mr. Timothy J    male  54.0      0
10                   Sandstrom, Miss. Marguerite Rut  female   4.0      1
11                          Bonnell, Miss. Elizabeth  female  58.0      0

    Parch    Ticket     Fare Cabin Embarked
1       0  PC 17599  71.2833   C85        C
3       0    113803  53.1000  C123        S
6       0     17463  51.8625   E46        S
10      1   PP 9549  16.7000    G6        S
11      0    113783  26.5500  C103        S

Dataset after Filling Missing Values with Mean for Numerical Columns:
   PassengerId  Survived  Pclass  \
0            1         0       3
1            2         1       1
```

```
2            3         1       3
3            4         1       1
4            5         0       3

                                                 Name     Sex   Age  SibSp  \
0                             Braund, Mr. Owen Harris    male  22.0      1
1  Cumings, Mrs. John Bradley (Florence Briggs Th…  female  38.0      1
2                              Heikkinen, Miss. Laina  female  26.0      0
3        Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
4                            Allen, Mr. William Henry    male  35.0      0

   Parch            Ticket     Fare Cabin Embarked
0      0         A/5 21171   7.2500   NaN        S
1      0          PC 17599  71.2833   C85        C
2      0  STON/O2. 3101282   7.9250   NaN        S
3      0            113803  53.1000  C123        S
4      0            373450   8.0500   NaN        S

Dataset after Forward Fill:
   PassengerId  Survived  Pclass  \
0            1         0       3
1            2         1       1
2            3         1       3
3            4         1       1
4            5         0       3

                                                 Name     Sex   Age  SibSp  \
0                             Braund, Mr. Owen Harris    male  22.0      1
1  Cumings, Mrs. John Bradley (Florence Briggs Th…  female  38.0      1
2                              Heikkinen, Miss. Laina  female  26.0      0
3        Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
4                            Allen, Mr. William Henry    male  35.0      0

   Parch            Ticket     Fare Cabin Embarked
0      0         A/5 21171   7.2500   NaN        S
1      0          PC 17599  71.2833   C85        C
2      0  STON/O2. 3101282   7.9250   C85        S
3      0            113803  53.1000  C123        S
4      0            373450   8.0500  C123        S

Dataset after Backward Fill:
   PassengerId  Survived  Pclass  \
0            1         0       3
1            2         1       1
2            3         1       3
3            4         1       1
4            5         0       3
```

```
                                                Name     Sex   Age  SibSp  \
0                            Braund, Mr. Owen Harris    male  22.0      1
1  Cumings, Mrs. John Bradley (Florence Briggs Th…  female  38.0      1
2                             Heikkinen, Miss. Laina  female  26.0      0
3       Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
4                           Allen, Mr. William Henry    male  35.0      0

   Parch            Ticket     Fare Cabin Embarked
0      0         A/5 21171   7.2500   C85        S
1      0          PC 17599  71.2833   C85        C
2      0  STON/O2. 3101282   7.9250  C123        S
3      0            113803  53.1000  C123        S
4      0            373450   8.0500   E46        S

Dataset after Interpolation for Numerical Columns:
   PassengerId  Survived  Pclass  \
0            1         0       3
1            2         1       1
2            3         1       3
3            4         1       1
4            5         0       3

                                                Name     Sex   Age  SibSp  \
0                            Braund, Mr. Owen Harris    male  22.0      1
1  Cumings, Mrs. John Bradley (Florence Briggs Th…  female  38.0      1
2                             Heikkinen, Miss. Laina  female  26.0      0
3       Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
4                           Allen, Mr. William Henry    male  35.0      0

   Parch            Ticket     Fare Cabin Embarked
0      0         A/5 21171   7.2500   NaN        S
1      0          PC 17599  71.2833   C85        C
2      0  STON/O2. 3101282   7.9250   NaN        S
3      0            113803  53.1000  C123        S
4      0            373450   8.0500   NaN        S
```

6 6. How can you detect and remove duplicate entries in a big dataset?

[6]:
```python
import pandas as pd

# Load the dataset
df = pd.read_csv('titanic.csv')  # Replace with the path to your dataset

# 1. Detect duplicates
print("Detecting Duplicates:")
```

```
duplicates = df[df.duplicated()]
print(duplicates)  # Prints duplicate rows

# 2. Count the number of duplicate rows
duplicate_count = df.duplicated().sum()
print(f"\nTotal duplicate rows: {duplicate_count}")

# 3. Remove duplicates
df_no_duplicates = df.drop_duplicates()
print("\nDataset after Removing Duplicates:")
print(df_no_duplicates.head())

# 4. Remove duplicates but keep the last occurrence (if you prefer)
df_no_duplicates_last = df.drop_duplicates(keep='last')
print("\nDataset after Removing Duplicates (Keeping Last Occurrence):")
print(df_no_duplicates_last.head())

# 5. Remove duplicates based on specific columns
df_no_duplicates_columns = df.drop_duplicates(subset=['Age', 'Fare'])
print("\nDataset after Removing Duplicates based on Specific Columns:")
print(df_no_duplicates_columns.head())
```

```
Detecting Duplicates:
Empty DataFrame
Columns: [PassengerId, Survived, Pclass, Name, Sex, Age, SibSp, Parch, Ticket,
Fare, Cabin, Embarked]
Index: []

Total duplicate rows: 0

Dataset after Removing Duplicates:
   PassengerId  Survived  Pclass  \
0            1         0       3
1            2         1       1
2            3         1       3
3            4         1       1
4            5         0       3

                                                Name     Sex   Age  SibSp  \
0                            Braund, Mr. Owen Harris    male  22.0      1
1  Cumings, Mrs. John Bradley (Florence Briggs Th…  female  38.0      1
2                             Heikkinen, Miss. Laina  female  26.0      0
3       Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
4                           Allen, Mr. William Henry    male  35.0      0

   Parch            Ticket     Fare Cabin Embarked
0      0         A/5 21171   7.2500   NaN        S
1      0          PC 17599  71.2833   C85        C
```

```
2      0  STON/O2. 3101282   7.9250   NaN        S
3      0            113803  53.1000  C123        S
4      0            373450   8.0500   NaN        S

Dataset after Removing Duplicates (Keeping Last Occurrence):
   PassengerId  Survived  Pclass  \
0            1         0       3
1            2         1       1
2            3         1       3
3            4         1       1
4            5         0       3

                                                 Name     Sex   Age  SibSp  \
0                             Braund, Mr. Owen Harris    male  22.0      1
1  Cumings, Mrs. John Bradley (Florence Briggs Th…  female  38.0      1
2                              Heikkinen, Miss. Laina  female  26.0      0
3        Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
4                            Allen, Mr. William Henry    male  35.0      0

   Parch            Ticket     Fare Cabin Embarked
0      0         A/5 21171   7.2500   NaN        S
1      0          PC 17599  71.2833   C85        C
2      0  STON/O2. 3101282   7.9250   NaN        S
3      0            113803  53.1000  C123        S
4      0            373450   8.0500   NaN        S

Dataset after Removing Duplicates based on Specific Columns:
   PassengerId  Survived  Pclass  \
0            1         0       3
1            2         1       1
2            3         1       3
3            4         1       1
4            5         0       3

                                                 Name     Sex   Age  SibSp  \
0                             Braund, Mr. Owen Harris    male  22.0      1
1  Cumings, Mrs. John Bradley (Florence Briggs Th…  female  38.0      1
2                              Heikkinen, Miss. Laina  female  26.0      0
3        Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
4                            Allen, Mr. William Henry    male  35.0      0

   Parch            Ticket     Fare Cabin Embarked
0      0         A/5 21171   7.2500   NaN        S
1      0          PC 17599  71.2833   C85        C
2      0  STON/O2. 3101282   7.9250   NaN        S
3      0            113803  53.1000  C123        S
4      0            373450   8.0500   NaN        S
```

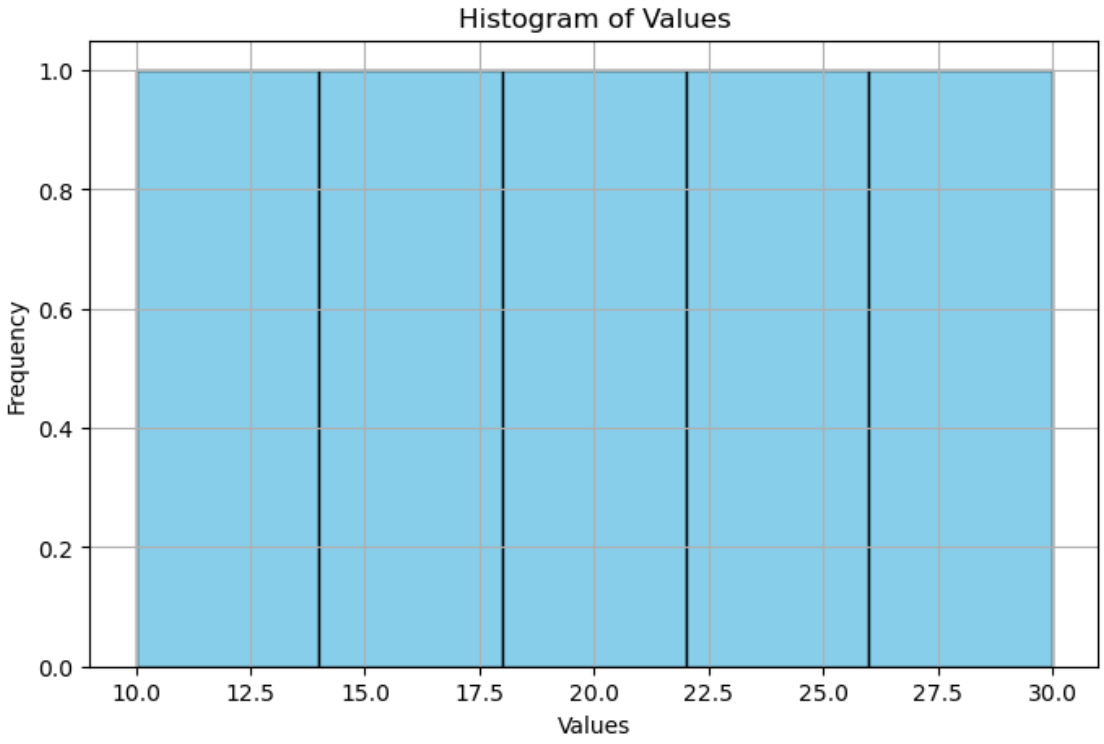
7 7. How can you implement and tune a Random Forest Regression model for housing price prediction?

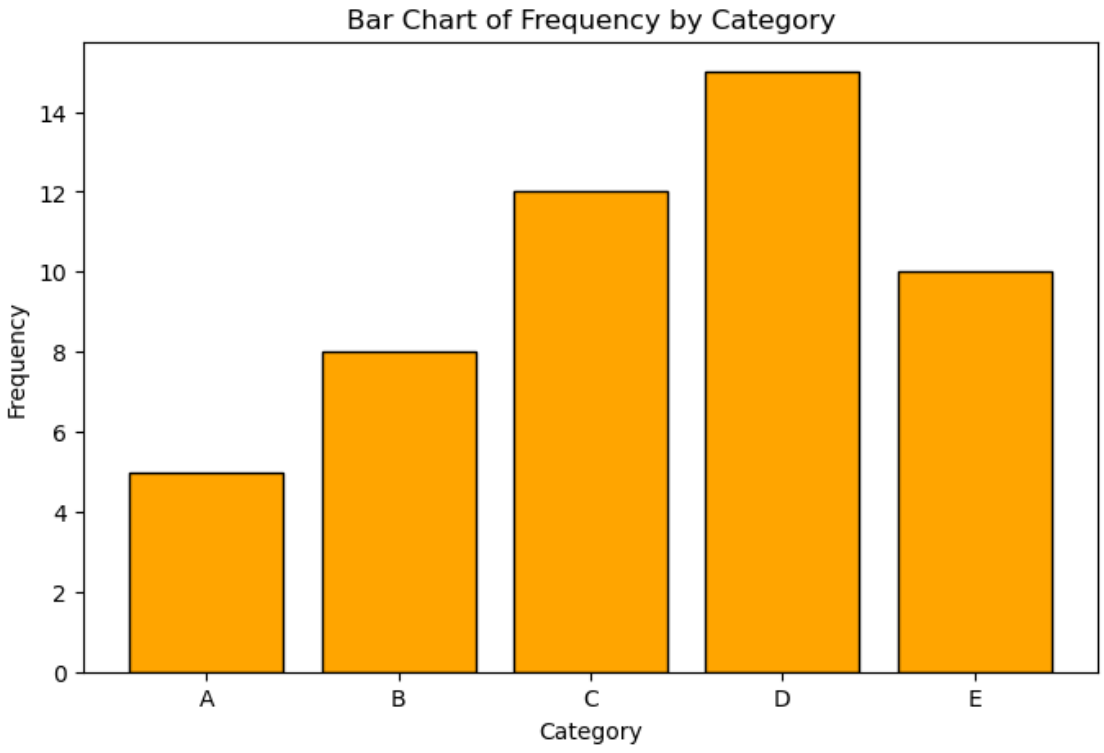
[ ]:
```
import pandas as pd
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import GridSearchCV, train_test_split
from sklearn.metrics import mean_squared_error
import numpy as np

# Load and preprocess the data
data = pd.read_csv('housing.csv').dropna()
data = pd.get_dummies(data, drop_first=True)

# Features and target
X = data.drop(['Price'], axis=1)
y = data['Price']

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)

# Initial Random Forest model
forest = RandomForestRegressor(random_state=42)
forest.fit(X_train, y_train)

# Evaluate the initial model
y_pred = forest.predict(X_test)
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
print(f"Initial model RMSE: {rmse:.2f}")

# Hyperparameter tuning with GridSearchCV
param_grid = {
    "n_estimators": [100, 200, 300],
    "max_features": [6, 8, 10],
}

grid_search = GridSearchCV(forest, param_grid, cv=5,␣
 ↪scoring="neg_mean_squared_error", n_jobs=-1)
grid_search.fit(X_train, y_train)

# Best model evaluation
best_forest_model = grid_search.best_estimator_
y_pred_best = best_forest_model.predict(X_test)
rmse_best = np.sqrt(mean_squared_error(y_test, y_pred_best))
print(f"Best model RMSE: {rmse_best:.2f}")
print(f"Best hyperparameters from GridSearchCV: {grid_search.best_params_}")
```

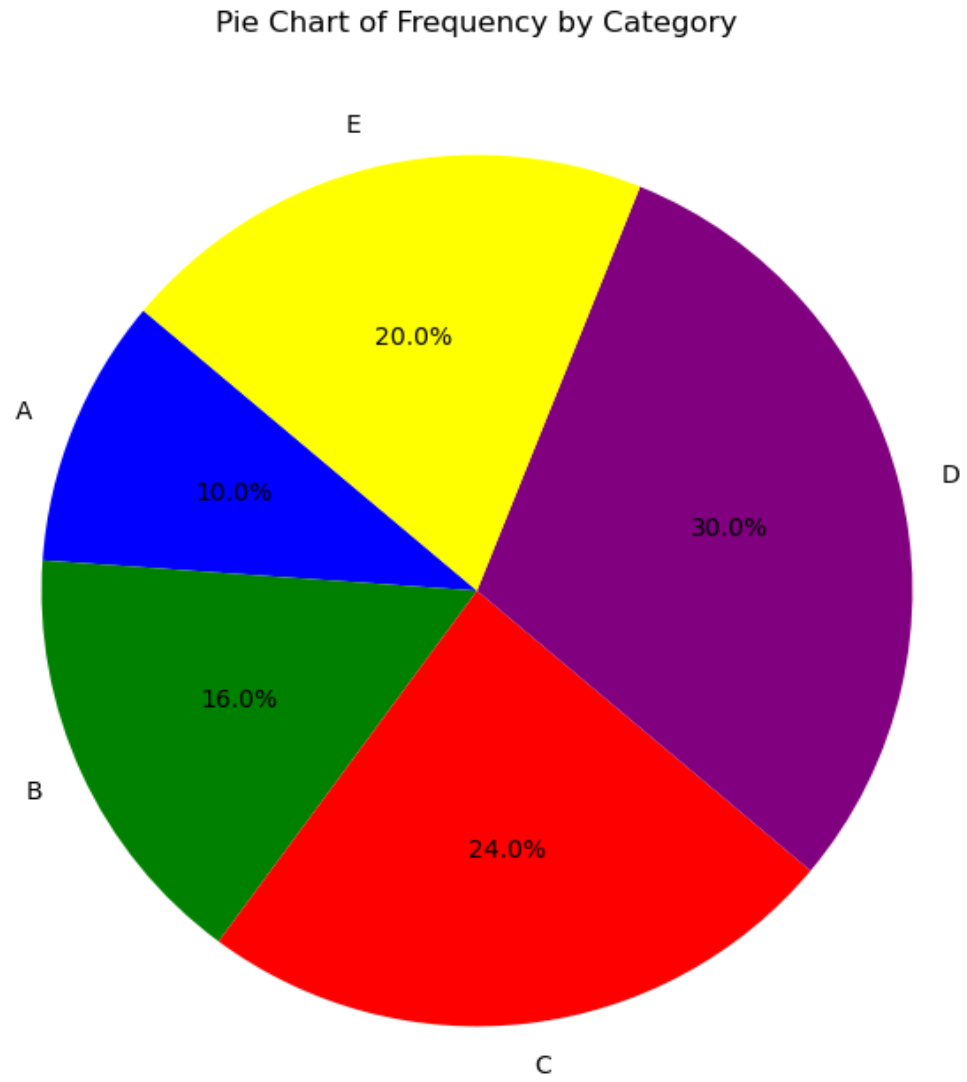
```
Initial model RMSE: 120501.35
```

8 8. Plot the histogram,bar chart and pie chart on a sample data set

[1]:
```
import pandas as pd
import matplotlib.pyplot as plt

# Sample dataset
data = {
    'Category': ['A', 'B', 'C', 'D', 'E'],
    'Values': [10, 15, 20, 25, 30],
    'Frequency': [5, 8, 12, 15, 10]
}

df = pd.DataFrame(data)

# Histogram: Distribution of 'Values'
plt.figure(figsize=(8, 5))
plt.hist(df['Values'], bins=5, color='skyblue', edgecolor='black')
plt.title('Histogram of Values')
plt.xlabel('Values')
plt.ylabel('Frequency')
plt.grid(True)
plt.show()

# Bar Chart: Frequency by Category
plt.figure(figsize=(8, 5))
plt.bar(df['Category'], df['Frequency'], color='orange', edgecolor='black')
plt.title('Bar Chart of Frequency by Category')
plt.xlabel('Category')
plt.ylabel('Frequency')
plt.show()

# Pie Chart: Proportion of Categories
plt.figure(figsize=(8, 8))
plt.pie(df['Frequency'], labels=df['Category'], autopct='%1.1f%%',␣
 ↪startangle=140, colors=['blue', 'green', 'red', 'purple', 'yellow'])
plt.title('Pie Chart of Frequency by Category')
plt.show()
```


Histogram of Values
Frequency
Values
0.0
0.2
0.4
0.6
0.8
1.0
10.0
12.5
15.0
17.5
20.0
22.5
25.0
27.5
30.0


Bar Chart of Frequency by Category
Frequency
Category
0
2
4
6
8
10
12
14
A
B
C
D
E



9 9. Implement Linear and logistic Regression on a sample dataset

[2]:
```
import numpy as np
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score

# Sample data for Linear Regression
data = {
    'Area (sq ft)': [500, 1000, 1500, 2000, 2500],
```

```
    'Bedrooms': [1, 2, 3, 3, 4],
    'Price': [100000, 200000, 300000, 400000, 500000]
}
df = pd.DataFrame(data)

# Features and target
X = df[['Area (sq ft)', 'Bedrooms']]
y = df['Price']

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)

# Linear Regression model
linear_model = LinearRegression()
linear_model.fit(X_train, y_train)

# Predictions
y_pred = linear_model.predict(X_test)

# Evaluation
print("Linear Regression Results:")
print(f"Mean Squared Error: {mean_squared_error(y_test, y_pred):.2f}")
print(f"R^2 Score: {r2_score(y_test, y_pred):.2f}")
print(f"Coefficients: {linear_model.coef_}")
print(f"Intercept: {linear_model.intercept_}")
```

```
Linear Regression Results:
Mean Squared Error: 0.00
R^2 Score: nan
Coefficients: [2.00000000e+02 1.14339053e-11]
Intercept: 5.820766091346741e-11

C:\Users\aishwarya\anaconda3\Lib\site-
packages\sklearn\metrics\_regression.py:1266: UndefinedMetricWarning: R^2 score
is not well-defined with less than two samples.
  warnings.warn(msg, UndefinedMetricWarning)
```

10 10. How do you use Python to create lagfeatures for time-series datasets.

[3]:
```
import pandas as pd

# Sample time-series data
data = {
    'Date': pd.date_range(start='2021-01-01', periods=10, freq='D'),
    'Sales': [200, 220, 250, 270, 300, 320, 350, 380, 400, 420]
```

```
}

# Create a DataFrame
df = pd.DataFrame(data)
df.set_index('Date', inplace=True)

print("Original Data:")
print(df)

# Creating lag features
df['Lag_1'] = df['Sales'].shift(1)  # Lag of 1 day
df['Lag_2'] = df['Sales'].shift(2)  # Lag of 2 days
df['Lag_3'] = df['Sales'].shift(3)  # Lag of 3 days

# Adding rolling average as an example of another feature
df['Rolling_Avg_3'] = df['Sales'].rolling(window=3).mean()

print("\nData with Lag Features:")
print(df)
```

```
Original Data:
            Sales
Date
2021-01-01    200
2021-01-02    220
2021-01-03    250
2021-01-04    270
2021-01-05    300
2021-01-06    320
2021-01-07    350
2021-01-08    380
2021-01-09    400
2021-01-10    420

Data with Lag Features:
            Sales  Lag_1  Lag_2  Lag_3  Rolling_Avg_3
Date
2021-01-01    200    NaN    NaN    NaN            NaN
2021-01-02    220  200.0    NaN    NaN            NaN
2021-01-03    250  220.0  200.0    NaN     223.333333
2021-01-04    270  250.0  220.0  200.0     246.666667
2021-01-05    300  270.0  250.0  220.0     273.333333
2021-01-06    320  300.0  270.0  250.0     296.666667
2021-01-07    350  320.0  300.0  270.0     323.333333
2021-01-08    380  350.0  320.0  300.0     350.000000
2021-01-09    400  380.0  350.0  320.0     376.666667
2021-01-10    420  400.0  380.0  350.0     400.000000
```