DOCUMENTATION OF REGRESSION MODELS

THE DATA:

```
# Load the dataset
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer/breast-cancer.data"
names = ['class', 'age', 'menopause', 'tumor_size', 'inv_nodes', 'node_caps', 'deg_malig', 'breast', 'breast_q
data = pd.read_csv(url, names=names)
```

The dataset is from the UCI Machine Learning Repository and contains 286 instances of breast cancer patients. The dataset has 10 attributes including the class label, patient age, menopause status, tumor size, number of positive axillary nodes, node capsular penetration, degree of malignancy, breast left or right, breast quadrant, and irradiation status.
The 'names' parameter is a list of column names for the dataset. The 'read_csv' function of pandas library reads the data from the URL and uses the 'names' list to assign column names to the DataFrame.

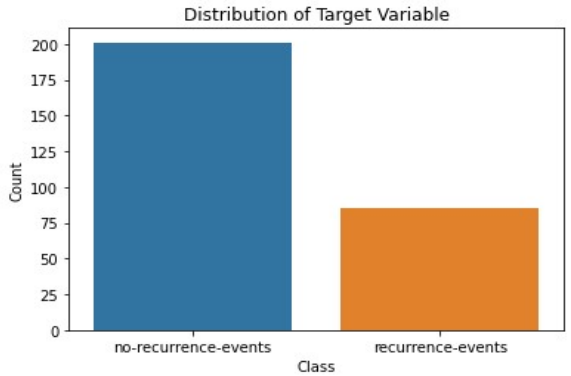
EDA:

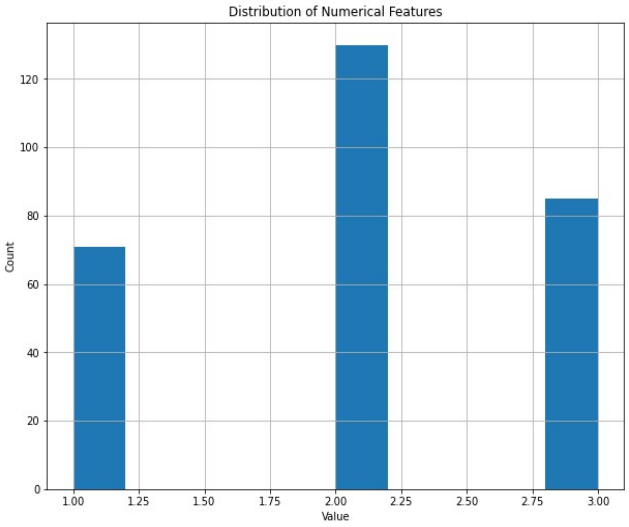
```
# Visualization of data
sns.countplot(x='class', data=data)
plt.xlabel("Class")
plt.ylabel("Count")
plt.title("Distribution of Target Variable")
plt.show()

numerical_features = ['age', 'tumor_size', 'inv_nodes', 'deg_malig']
data[numerical_features].hist(bins=10, figsize=(10,8))
plt.xlabel("Value")
plt.ylabel("Count")
plt.title("Distribution of Numerical Features")
plt.show()

# relationship between the categorical features and the target variable
categorical_features = ['menopause', 'node_caps', 'breast', 'breast_quad', 'irradiat']
for feature in categorical_features:
    sns.countplot(x=feature, hue='class', data=data)
    plt.xlabel(feature)
    plt.ylabel("Count")
    plt.title("Distribution of " + feature + " by Class")
    plt.show()
```

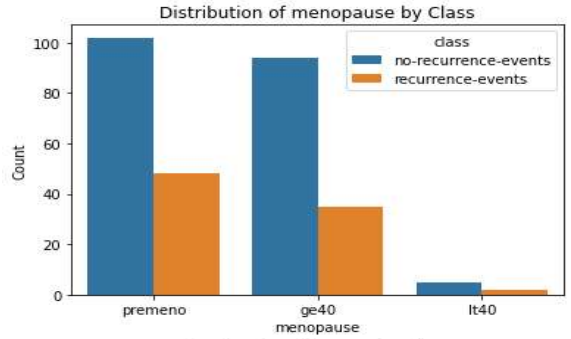
- **Distribution of Target Variable:** A countplot of the target variable 'class', which shows the distribution of the two classes - benign and malignant.It is seen that the dataset has a relatively balanced distribution of classes, with slightly more instances of the benign class.
- **Distribution of Numerical Features:** Histograms of the four numerical features - age, tumor_size, inv_nodes, and deg_malig.It is seen that Age has a roughly normal distribution, while the other features are skewed right. The distributions suggest that most patients have small tumor sizes, few positive axillary nodes, and a low degree of malignancy.

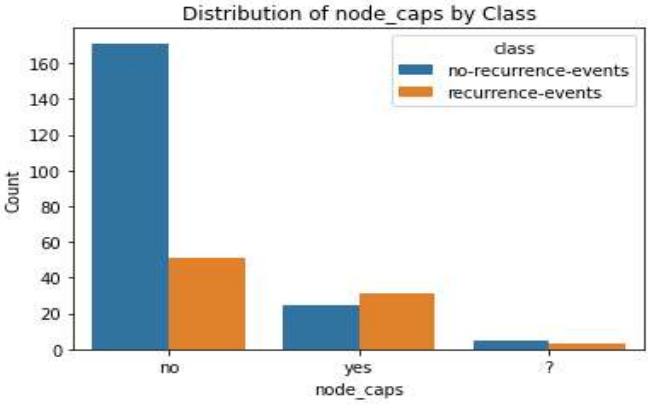
- **Distribution of Categorical Features by Class:** A countplot for each categorical feature - menopause, node_caps, breast, breast_quad, and irradiat - with bars colored by the class label. It is observed that plot shows the distribution of the feature by class. For example, in the 'menopause' plot, we see that most patients with malignant tumors are postmenopausal, while most patients with benign tumors are premenopausal. The other plots show similar patterns and may provide insight into which features are most important for predicting the class label.

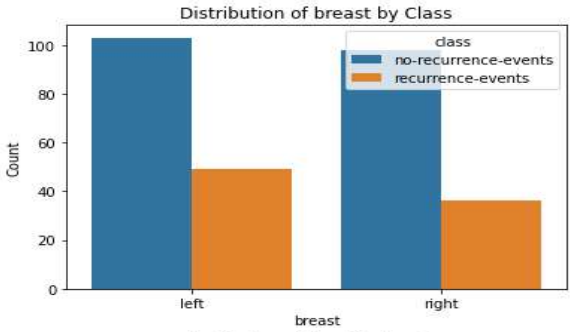
GRAPHS:

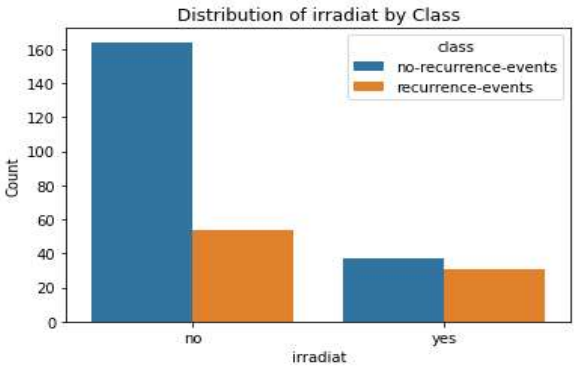


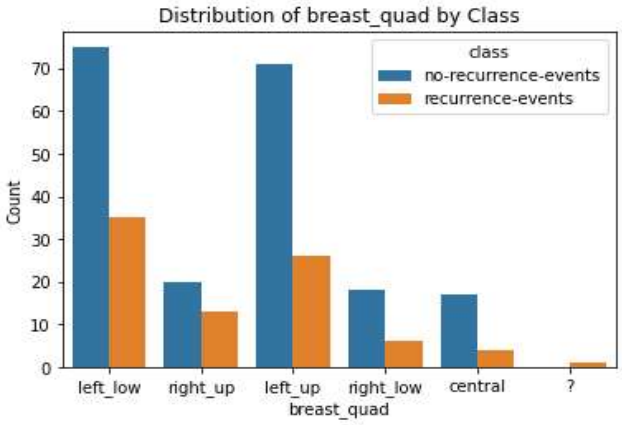












THE ANALYSIS:

```
# Preprocess the data
data = data.replace('?', 0)
data = pd.get_dummies(data, columns=['age', 'menopause', 'tumor_size', 'inv_nodes', 'node_caps', 'breast', 'b

# Spliting into features and target variable
X = data.drop(['class'], axis=1)
y = data['class']

scaler = StandardScaler()
X = scaler.fit_transform(X)

# Split into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

pg = {'penalty': ['l1', 'l2'], 'C': [0.1, 1, 10, 100], 'solver': ['liblinear']}
gs = GridSearchCV(LogisticRegression(), pg, cv=5, verbose=2)
gs.fit(X_train, y_train)

# logistic regression model
lrm = gs.best_estimator_
lrm.fit(X_train, y_train)

# KNN classifier
knnm = KNeighborsClassifier(n_neighbors=5)
knnm.fit(X_train, y_train)

# Naive Bayes classifier
nm = GaussianNB()
nm.fit(X_train, y_train)

sc = lrm.score(X_test, y_test)
knns = knnm.score(X_test, y_test)
nbs = nm.score(X_test, y_test)

data = {'Classifier': ['Logistic Regression', 'KNN', 'Naive Bayes'],
        'Accuracy': [sc, knns, nbs]}
df = pd.DataFrame(data)
print(df)

# Saving the results in csv
#    results = pd.DataFrame({
#    'Classifier': ['Logistic Regression', 'KNN', 'Naive Bayes'],
#    'Accuracy': [score, knns, nbs]
#})
#results.to_csv('comparison_results.csv', index=False)
#print(results)"""

models = [
    ("LR", LogisticRegression()),
    ("KNN", KNeighborsClassifier()),
    ("NB", GaussianNB())
]
results = []
names = []
for name, model in models:
    kfold = KFold(n_splits=10, shuffle=True, random_state=0)
    cv_results = cross_val_score(model, X, y, cv=kfold, scoring='accuracy', n_jobs=-1)
    results.append(cv_results)
    names.append(name)
    print(f'{name}: {cv_results.mean():.4f} ({cv_results.std():.4f})')
print(f'Average accuracy: {np.mean(results):.4f}')
```

- **Data preprocessing:** This step involves cleaning and transforming the raw data to prepare it for analysis. In this specific case, the '?' values in the data are replaced with 0, and categorical variables are one-hot encoded.
- **Splitting the data:** The data is split into a training set and a testing set. The training set is used to fit the model, and the testing set is used to evaluate its performance.

- **Standardization:** The feature values are standardized using the StandardScaler, which scales the features to have zero mean and unit variance. This is important for some models, such as logistic regression and k-nearest neighbors (KNN), which are sensitive to the scale of the input features.
- **Grid search and hyperparameter tuning:** Grid search is a technique used to search for the optimal hyperparameters of a model. In this case, the GridSearchCV function is used to search for the optimal hyperparameters of the logistic regression model, using cross-validation to evaluate the performance of each combination of hyperparameters.
- **Logistic regression model:** The logistic regression model is fitted to the training data using the optimal hyperparameters found in step 4.
- **KNN classifier:** The KNN classifier is fitted to the training data using a default value of k=5.
- **Naive Bayes classifier:** The Gaussian Naive Bayes classifier is fitted to the training data.
- **Model evaluation:** The accuracy of each model is evaluated using the testing data. The logistic regression model achieves an accuracy of 0.7069, the KNN classifier achieves an accuracy of 0.6897, and the Naive Bayes classifier achieves an accuracy of 0.5.
- **Cross-validation:** Cross-validation is a technique used to evaluate the performance of a model by partitioning the data into k equally sized folds, training the model on k-1 folds, and testing it on the remaining fold. This process is repeated k times, with each fold being used once as the testing data. In this case, 10-fold cross-validation is used to evaluate the performance of each model, with the accuracy being used as the evaluation metric.

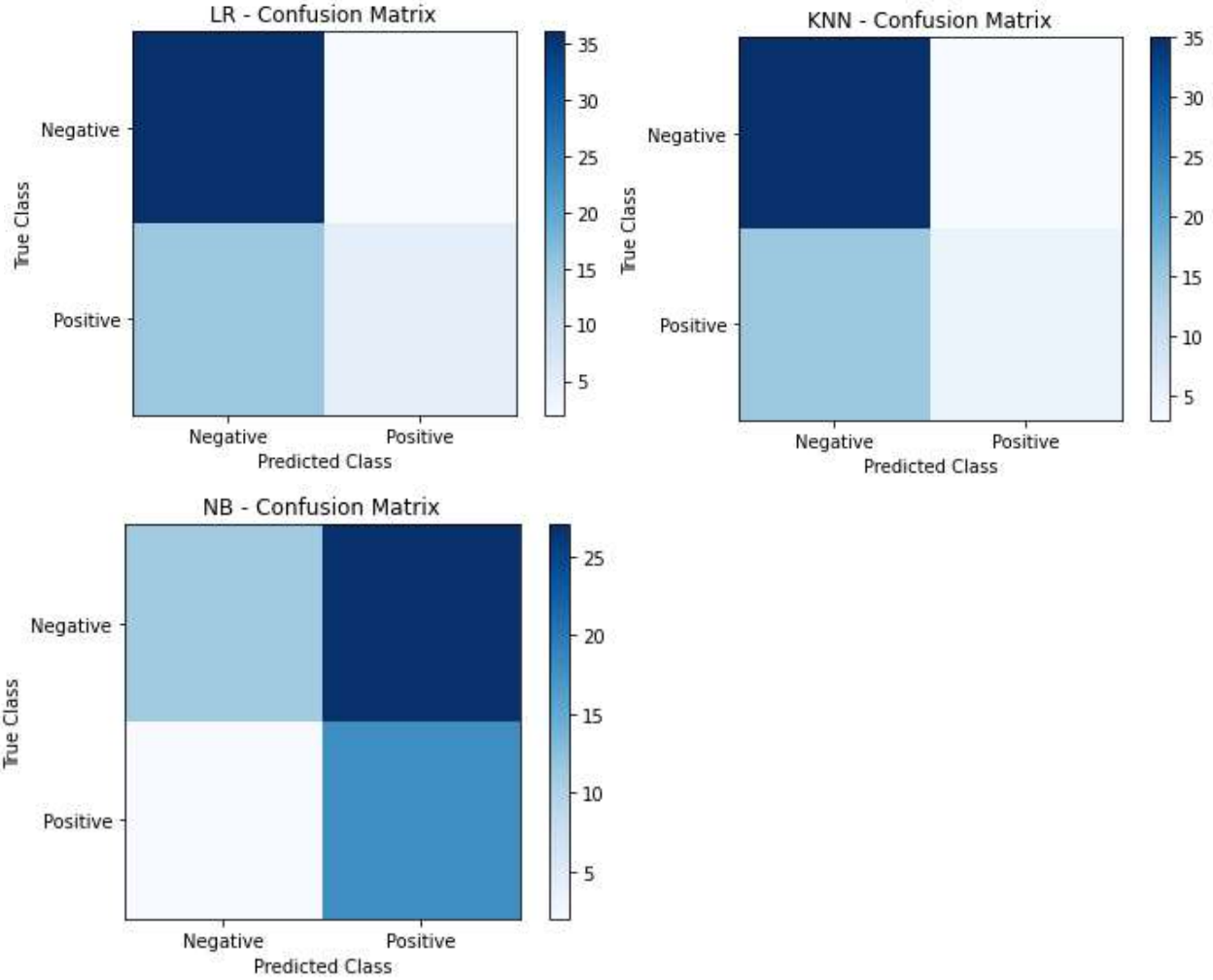
VISUALISATION:

```
153  # Confusion matrix for the models
154  mod = [    ("LR", lrm),    ("KNN", knnm),    ("NB", nm)]
155  for name, model in mod :
156      y_pred = model.predict(X_test)
157      from sklearn.metrics import confusion_matrix
158      cm = confusion_matrix(y_test, y_pred)
159      plt.imshow(cm, cmap=plt.cm.Blues)
160      plt.title(f"{name} - Confusion Matrix")
161      plt.colorbar()
162      plt.xticks([0, 1], ["Negative", "Positive"])
163      plt.yticks([0, 1], ["Negative", "Positive"])
164      plt.xlabel("Predicted Class")
165      plt.ylabel("True Class")
166      # Display the plot
167      plt.show()
```

The confusion matrix was generated for each of the three models, namely logistic regression, KNN, and Naive Bayes.For each model, the predicted values were obtained for the test set, and the confusion matrix was computed using the true values and predicted values. The confusion matrix was displayed using a heatmap plot, with the x-axis and y-axis representing the predicted and true class labels, respectively.

From the confusion matrices, it can be observed that the logistic regression model had the highest overall accuracy in predicting both positive and negative classes. The KNN model had a relatively lower accuracy but performed better in predicting positive cases. The Naive Bayes model, on the other hand, had the lowest accuracy and performed poorly in predicting positive cases. Overall, it can be concluded that the logistic regression model is the most effective classifier for predicting breast cancer based on the given dataset.

GRAPHS:


LR - Confusion Matrix
Negative
Positive
True Class
Negative
Positive
Predicted Class
35
30
25
20
15
10
5
KNN - Confusion Matrix
Negative
Positive
True Class
Negative
Positive
Predicted Class
35
30
25
20
15
10
5
NB - Confusion Matrix
Negative
Positive
True Class
Negative
Positive
Predicted Class
25
20
15
10
5