Predicting Health Outcomes of Horses with Machine Learning

Context

Welcome to the **Scaler Health Analytics** team! Our objective is to leverage advanced analytics to predict health outcomes in horses, enabling veterinarians to make more informed decisions that can improve treatment strategies and overall horse health management. Utilizing machine learning techniques, particularly classification algorithms, you will analyze clinical data from horses to predict various health outcomes.

Dataset Description

You have been provided with a comprehensive dataset containing clinical records of horses treated in a veterinary hospital. The data encapsulates a variety of clinical attributes that offer insights into the general health and medical condition of each horse. Each record in the dataset is described by the following features:

Clinical Attributes:

- **id:** Unique identifier for each horse.
- **surgery:** Whether the horse had surgery (Yes/No).
- **age:** Age of the horse.
- **hospital_number:** Unique number assigned to the horse at the hospital.
- **rectal_temp:** Rectal temperature of the horse.
- **pulse:** Pulse rate in beats per minute.
- **respiratory_rate:** Respiratory rate in breaths per minute.
- **temp_of_extremities:** Temperature of extremities (a possible indicator of shock).
- **peripheral_pulse:** Quality of the peripheral pulse.
- **mucous_membrane:** Color of the mucous membranes, which can indicate blood circulation quality.
- **capillary_refill_time:** Time taken for color to return to mucous membrane after pressure is applied.
- **pain:** Horse's pain level (graded).
- **peristalsis:** Intestinal activity observed.
- **abdominal_distention:** Any distention of the abdomen.
- **nasogastric_tube:** Whether a nasogastric tube has been placed.
- **nasogastric_reflux:** Any nasogastric reflux noted.
- **nasogastric_reflux_ph:** pH of the nasogastric reflux.
- **rectal_exam_feces:** Findings of the rectal examination of feces.
- **abdomen:** Detailed examination findings of the abdomen.
- **packed_cell_volume:** Packed cell volume, indicating hydration status and blood loss.
- **total_protein:** Total protein levels in blood.
- **abdomo_appearance:** Appearance of abdominal fluid.
- **abdomo_protein:** Protein level in abdominal fluid.
- **surgical_lesion:** Presence of surgical lesions.
- **lesion_1, lesion_2, lesion_3:** Type and location of lesions identified.
- **cp_data:** Clinical pathology data.
- **outcome:** Health outcome of the horse (e.g., recovered, euthanized, died).

Your task is to use machine learning models to predict the 'outcome' for each horse and identify key predictors of health outcomes. This project will contribute significantly to improving the predictive models used in veterinary practices.

```
!wget https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/072/101/original/health_outcome_dataset.csv
```

```
--2025-07-31 13:17:17--  https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/072/101/original/health_outcome_da
Resolving d2beiqkhq929f0.cloudfront.net (d2beiqkhq929f0.cloudfront.net)... 18.172.139.210, 18.172.139.61, 18.172.139.94, .
Connecting to d2beiqkhq929f0.cloudfront.net (d2beiqkhq929f0.cloudfront.net)|18.172.139.210|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 229201 (224K) [text/plain]
Saving to: ‘health_outcome_dataset.csv.3’

health_outcome_data 100%[===================>] 223.83K  --.-KB/s    in 0.04s

2025-07-31 13:17:17 (6.03 MB/s) - ‘health_outcome_dataset.csv.3’ saved [229201/229201]
```

```
import warnings

# Ignore all warnings
warnings.filterwarnings("ignore")
```

```
import pandas as pd

df = pd.read_csv("health_outcome_dataset.csv")
df.head()
```

	id	surgery	age	hospital_number	rectal_temp	pulse	respiratory_rate	temp_of_extremities	peripheral_pulse	mucous_
0	0	yes	adult	530001	38.1	132.0	24.0	cool	reduced	dar
1	1	yes	adult	533836	37.5	88.0	12.0	cool	normal	pal
2	2	yes	adult	529812	38.3	120.0	28.0	cool	reduced	
3	3	yes	adult	5262541	37.1	72.0	30.0	cold	reduced	
4	4	no	adult	5299629	38.0	52.0	48.0	normal	normal	no

5 rows × 29 columns

```
df.isna().sum()
```

	0
id	0
surgery	0
age	0
hospital_number	0
rectal_temp	0
pulse	0
respiratory_rate	0
temp_of_extremities	39
peripheral_pulse	60
mucous_membrane	21
capillary_refill_time	6
pain	44
peristalsis	20
abdominal_distention	23
nasogastric_tube	80
nasogastric_reflux	21
nasogastric_reflux_ph	0
rectal_exam_feces	190
abdomen	213
packed_cell_volume	0
total_protein	0
abdomo_appearance	48
abdomo_protein	0
surgical_lesion	0
lesion_1	0
lesion_2	0
lesion_3	0
cp_data	0
outcome	0

dtype: int64

```
import pandas as pd
from sklearn.impute import SimpleImputer

df.drop('lesion_3', inplace = True, axis = 1)

# Create imputer objects
num_imputer = SimpleImputer(strategy='median')  # Imputer for numerical data
cat_imputer = SimpleImputer(strategy='most_frequent')  # Imputer for categorical data

# Define columns by type
num_cols = ['rectal_temp', 'pulse', 'respiratory_rate']  # Numerical columns
cat_cols = ['temp_of_extremities', 'peripheral_pulse', 'mucous_membrane', 'capillary_refill_time',
            'pain', 'peristalsis', 'abdominal_distention', 'nasogastric_tube',
            'nasogastric_reflux', 'abdomen', 'abdomo_appearance', 'rectal_exam_feces']  # Categorical columns

# Apply imputation
df[num_cols] = num_imputer.fit_transform(df[num_cols])
df[cat_cols] = cat_imputer.fit_transform(df[cat_cols])


df['outcome'].value_counts()
```

outcome	count
lived	574
died	410
euthanized	251

dtype: int64

Assesments for Naive Bayes-I

Pulse Rate Distribution

Context:

Analyzing the pulse rate distribution among horses with different health outcomes can provide critical insights into physiological states associated with various health conditions. This analysis can help veterinarians understand how pulse rate might correlate with the severity or type of condition a horse is facing.

Task:

Create a visual representation to analyze the pulse rate distribution across different health outcomes using the 'pulse' and 'outcome' columns in the dataset.

Instructions:

1. **Group and Count Data:** Use the dataset to group entries by 'outcome' and calculate the average 'pulse' for each outcome category.
2. **Visualize Data:** Generate a box plot to display the distribution of pulse rates for each health outcome category. This will help in visually comparing the pulse rate variability associated with different outcomes.
3. **Analyze Trends:** Examine the box plot to determine which health outcome is associated with the highest average pulse rate and which shows the most variability in pulse rates.

Question:

After analyzing the box plot representing the pulse rate distribution for different health outcomes, identify the correct statements regarding pulse rates.

Options:

A) The average pulse rate is highest among horses that lived, indicating recovery from potentially distressing conditions.

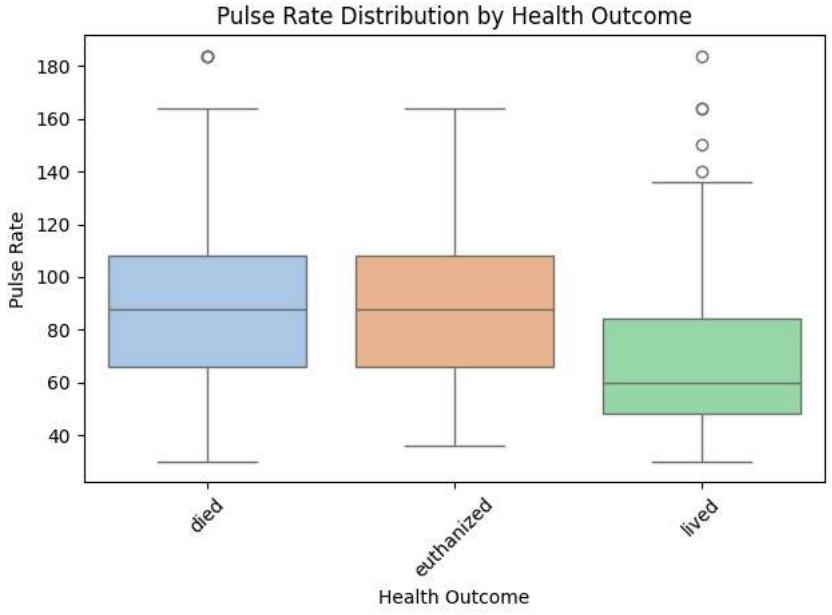
B) The average pulse rate is lowest for horses that were euthanized, suggesting less physiological distress before euthanasia.

C) The average pulse rate is highest among horses that died, suggesting a correlation between high pulse rates and critical health conditions leading to death.

D) The box plot shows no significant difference in the average pulse rates across different health outcomes.

```
import matplotlib.pyplot as plt
import seaborn as sns

# TODO: Group the dataframe by 'outcome' and calculate the average 'pulse' for each outcome category.
pulse_outcome_data = df.groupby('outcome')["pulse"].mean()  # Consider what function might summarize the data effectively.
print(pulse_outcome_data)

# Set a custom color palette (optional)
custom_palette = sns.color_palette("pastel")

# TODO: Utilize seaborn to create a box plot that shows variation within grouped data.
sns.boxplot(x="outcome", y="pulse", data=df, palette=custom_palette)
plt.title('Pulse Rate Distribution by Health Outcome')
plt.xlabel('Health Outcome')
plt.ylabel('Pulse Rate')
plt.xticks(rotation=45)
plt.tight_layout()  # Adjust layout to not cut off labels

plt.show()
```

```
outcome
died          90.207317
euthanized    86.573705
lived         68.918118
Name: pulse, dtype: float64
```



C) The average pulse rate is highest among horses that died, suggesting a correlation between high pulse rates and critical health conditions leading to death.

Rectal Temperature Correlation

Context:

Understanding how rectal temperature correlates with health outcomes in horses can provide valuable insights for veterinary treatment strategies. Rectal temperature is a critical clinical parameter and its deviations from the norm can indicate various health conditions.

Task:

Analyze the impact of different rectal temperature categories on health outcomes. The rectal temperatures will be categorized into hypothermia, normal, and fever to observe how these conditions correlate with health outcomes.

Instructions:

1. **Create Temperature Categories:** Add a new column to the dataset, 'temp_category', which categorizes rectal temperature into hypothermia (<37.5°C), normal (37.5-38.5°C), and fever (>38.5°C).
2. **Group Data by Temperature Category:** Group the dataset by 'temp_category' and 'outcome' to analyze the distribution of health outcomes within each temperature category.

3. **Visualize and Analyze Data:** Create a bar chart to visualize the count of each health outcome within the different temperature categories. This will help identify any significant correlations between rectal temperature and health outcomes.

Question:

After analyzing the bar chart which depicts the count of health outcomes within each rectal temperature category, identify the correct statement regarding the correlation between rectal temperature categories and health outcomes.

Options:

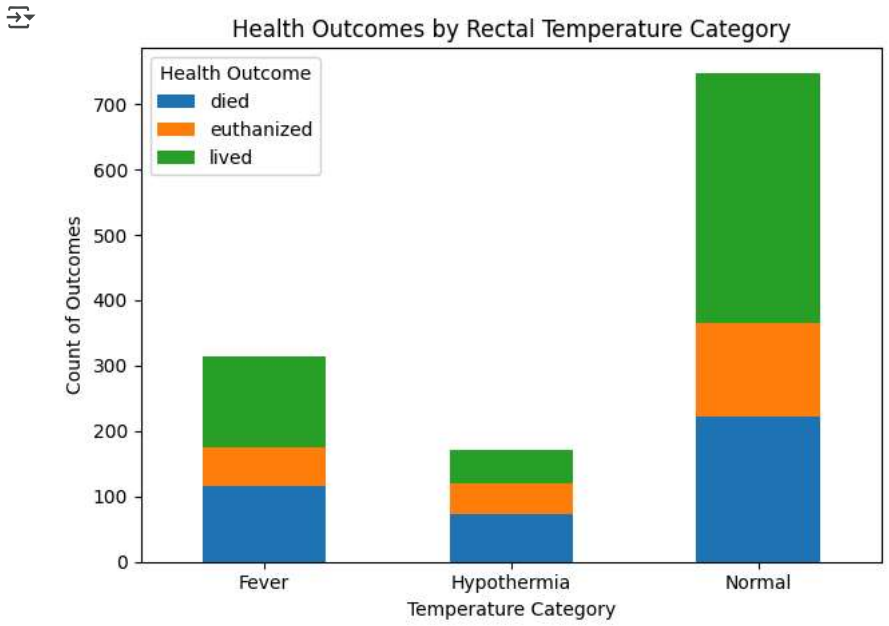
A) Horses with fever have the highest survival rate.

B) Normal rectal temperature is associated with the best health outcomes.

C) Horses with hypothermia are more likely to be euthanized.

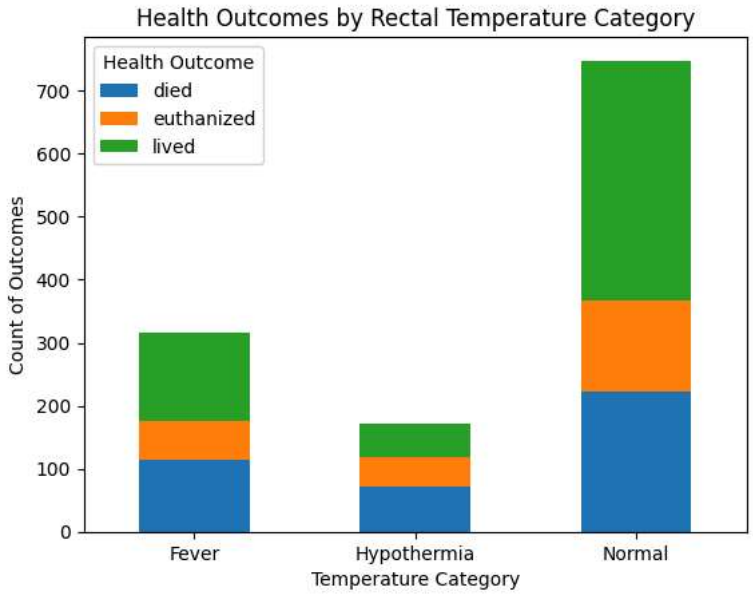
D) There is no significant relationship between rectal temperature categories and health outcomes.

```
# Define function to categorize rectal temperature
def categorize_temp(temp):
    # TODO: Return the category based on temperature values (use if-elif-else structure).
    if temp < 37.5:
        return "Hypothermia"
    elif 37.5 <= temp <= 38.5:
        return "Normal"
    else:
        return "Fever"

# Apply function to create a new column
df['temp_category'] = df['rectal_temp'].apply(categorize_temp)  # TODO: Choose the appropriate DataFrame method to apply a fun

# Grouping data by temperature category and outcome for plotting
grouped_data = df.groupby(['temp_category','outcome']).count()['id'].reset_index()  # TODO: Group by two columns and count a u
grouped_data.columns = ['Temperature Category', 'Outcome', 'Count']

# Create a pivot table for visualization
pivot_table = pd.pivot_table(grouped_data, values='Count', index='Temperature Category', columns='Outcome', fill_value=0)  # T

# Plotting the data
pivot_table.plot(kind='bar', stacked=True)
plt.title('Health Outcomes by Rectal Temperature Category')
plt.xlabel('Temperature Category')
plt.ylabel('Count of Outcomes')
plt.xticks(rotation=0)
plt.legend(title='Health Outcome')
plt.tight_layout()  # Adjust layout to not cut off labels

plt.show()
```



```
cross_tab=pd.crosstab(index=df["temp_category"],columns=df["outcome"])

cross_tab.plot(kind='bar', stacked=True)
plt.title('Health Outcomes by Rectal Temperature Category')
plt.xlabel('Temperature Category')
plt.ylabel('Count of Outcomes')
plt.xticks(rotation=0)
plt.legend(title='Health Outcome')
```

```
<matplotlib.legend.Legend at 0x796c1c20f3d0>
```



B) Normal rectal temperature is associated with the best health outcomes.

Naive Bayes Assumption and Its Impact

Context:

The Naive Bayes classifier, while simple and effective, assumes feature independence which may affect its application in complex datasets like "Predict Horse Health Outcomes."

Task:

Evaluate how the Naive Bayes assumption of feature independence affects its performance on a dataset with interdependent features such as surgery, age, pulse, and mucous membrane status.

Instructions:

1. **Review the Assumption:** Consider the independence assumption of Naive Bayes.
2. **Assess Potential Impacts:** Reflect on how this assumption might influence the model's effectiveness, especially where features might be interrelated.
3. **Select the Most Accurate Statement:** Choose the statement that best describes Naive Bayes' assumption and its impact on the dataset.

Statements:

1. **Naive Bayes treats all features as equally interdependent.** This is suited for datasets with strong feature correlations, potentially making it less effective for independent feature scenarios.
2. **Naive Bayes assumes independence among features.** This could pose challenges if features like pulse and mucous membrane status, which may be related, are not truly independent.
3. **Naive Bayes expects numerical features to behave like categorical variables.** This might complicate handling datasets with a mix of feature types.
4. **Naive Bayes requires features to be independent and may need preprocessing to handle correlations like those between pulse and mucous membrane status.**

Question:

Considering the assumption and potential impacts of Naive Bayes on the horse health outcomes dataset, which statement number is most accurate?

Option 2

Naive Bayes assumes independence among features. This could pose challenges if features like pulse and mucous membrane status, which may be related, are not truly independent.

Analyzing Pain and Euthanasia Decisions

Context:

Understanding the decision-making process in veterinary care, particularly euthanasia, can be crucial in managing horse health outcomes. Analyzing how pain levels influence these decisions is particularly significant.

Task:

Using Bayes' Theorem, calculate the probability that a horse is euthanized given it experiences severe pain and assess what this indicates about the role of pain in euthanasia decisions.

Instructions:

1. **Review Given Probabilities:**
    - P(Severe Pain) = 0.20
    - P(Euthanized) = 0.25
    - P(Severe Pain | Euthanized) = 0.50
2. **Calculate the Probability:** Use Bayes' Theorem to find P(Euthanized | Severe Pain).
3. **Interpret the Result:** Consider what the calculated probability implies about the impact of severe pain on euthanasia decisions.

Question:

Based on the calculation using Bayes' Theorem, what is the probability that a horse is euthanized given that it is experiencing severe pain? What does this probability suggest about the impact of severe pain on the decision to euthanize?

Options:

A) 0.40

B) 0.50

C) 0.75

D) 0.625

```
# Given probabilities
P_SeverePain = 0.20
P_Euthanized = 0.25
P_SeverePain_Euthanized = 0.50

# Bayes' Theorem calculation
P_Euthanized_SeverePain = P_SeverePain_Euthanized*P_Euthanized/P_SeverePain  # TODO: Calculate the probability using Bayes' Th
print("Probability of being euthanized given severe pain: {:.3f}".format(P_Euthanized_SeverePain))
```

```
Probability of being euthanized given severe pain: 0.625
```

Evaluating Feature Significance with Chi-square Test

Context:

In statistical analysis for data, the Chi-square test of independence is essential for determining the association between features. This test helps identify which features are statistically significant in relation to the outcome variable in a dataset.

Task:

Perform a Chi-square test of independence on each Categorical feature against the 'outcome' variable to determine their significance in predicting health outcomes of horses. Given the significance level of 0.05, identify which Categoricalfeature is not significantly

associated with the outcome.

Instructions:

1. **Perform Chi-square Tests:** Apply the Chi-square test to each Categorical feature against the 'outcome' variable.
2. **Interpret Results:** Evaluate the p-values to determine which Categorical feature(s) fail to show a significant association with the outcome, suggesting they may not be useful for predictive modeling in this context.

Question:

Based on a significance level of 0.05, which Categorical feature was found to be not significantly associated with the 'outcome' variable, therefore failing the Chi-square test?

Options:

A) temp_of_extremities

B) peripheral_pulse

C) pain

D) All columns passed

```
from scipy.stats import chi2_contingency  # TODO: Import the necessary function for performing the Chi-square test.

threshold = .05

print(f'{"Column":<25} | Test result')
print('----------------------------------------')

for column in cat_cols:
    # Create a contingency table
    contingency_table = pd.crosstab(df['outcome'], df[column])  # TODO: Create a contingency table from two categorical column

    # Perform the Chi-Square test
    chi2, p, dof, expected_freq = chi2_contingency(contingency_table)  # TODO: Apply the imported function to compute the Chi-

    print(f'{column:<25} |   ', '\033[32mPassed' if p < threshold else '\033[31mFailed', '\033[0m')
```

```
Column                    | Test result
----------------------------------------
temp_of_extremities       |    Passed
peripheral_pulse          |    Passed
mucous_membrane           |    Passed
capillary_refill_time     |    Passed
pain                      |    Passed
peristalsis               |    Passed
abdominal_distention      |    Passed
nasogastric_tube          |    Passed
nasogastric_reflux        |    Passed
abdomen                   |    Passed
abdomo_appearance         |    Passed
rectal_exam_feces         |    Passed
```

D) All columns passed

Data Preprocessing for Modelling

Context:

In preparation for building a Gaussian Naive Bayes model to predict horse health outcomes, the initial step involves preprocessing the dataset. This includes dropping less relevant columns and applying appropriate transformations to the remaining data.

Task:

Prepare the data by dropping specified columns and applying transformations suitable for a Gaussian Naive Bayes model.

Instructions:

1. **Drop Columns:**
    - Remove the 'id' columns from the dataset.
2. **Apply Transformations:**

- Use `StandardScaler` to normalize numerical data.
- Apply `OrdinalEncoder` to encode categorical data that handles unknown values.

3. **Split the Data:**
   - Use `train_test_split` with a `random_state=0` to divide the data into training and test sets, targeting 'outcome' as the label. `test_size = 0.2`

Question:

After preprocessing, which categorical feature ends up with the highest number of unique ordinal values?

Options:

A) temp_of_extremities

B) pain and mucous_membrane

C) cp_data and abdomen

D) nasogastric_reflux

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, OrdinalEncoder  # TODO: Import necessary preprocessing classes.
from sklearn.compose import ColumnTransformer

# Drop unnecessary columns
df_temp = df.drop(['id'], axis=1)

# Identify numerical and categorical columns
numerical_cols = df_temp.select_dtypes(include=['int', 'float']).columns.tolist()  # TODO: Specify data types to select numeri
categorical_cols = df_temp.select_dtypes(include=['object']).columns.tolist()  # TODO: Specify the data type for categorical(o

# Remove the target column from the feature lists
categorical_cols.remove('outcome')

# Prepare target and features
X = df_temp.drop('outcome', axis=1)
y = df_temp['outcome']

# Split the dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)  # TODO: Use the appropriate function

# Create transformers for numerical and categorical data
preprocessor = ColumnTransformer(
    transformers=[
        ('num', StandardScaler(), numerical_cols),  # TODO: Choose the correct transformer for numerical data.
        ('cat', OrdinalEncoder(), categorical_cols)  # TODO: Choose the correct transformer for categorical data that handles
    ])

# Apply preprocessing
X_train = preprocessor.fit_transform(X_train)  # TODO: Fit and Apply the preprocessor to training data.
X_test = preprocessor.transform(X_test)  # TODO: Apply the preprocessor to test data.


print(f'{"Column":<25} | Number of unique values')
print('------'*10)
for i in categorical_cols:
    print(f'{i:<25} |   ', f'\033[32m{df[i].unique()}' , '\033[0m')
```

```
Column                    | Number of unique values
------------------------------------------------------------
surgery                   |    ['yes' 'no']
age                       |    ['adult' 'young']
temp_of_extremities       |    ['cool' 'cold' 'normal' 'warm']
peripheral_pulse          |    ['reduced' 'normal' 'absent' 'increased']
mucous_membrane           |    ['dark_cyanotic' 'pale_cyanotic' 'pale_pink' 'normal_pink' 'bright_pink'
 'bright_red']
capillary_refill_time     |    ['more_3_sec' 'less_3_sec' '3']
pain                      |    ['depressed' 'mild_pain' 'extreme_pain' 'alert' 'severe_pain' 'slight']
peristalsis               |    ['absent' 'hypomotile' 'normal' 'hypermotile' 'distend_small']
abdominal_distention      |    ['slight' 'moderate' 'none' 'severe']
nasogastric_tube          |    ['slight' 'none' 'significant']
nasogastric_reflux        |    ['less_1_liter' 'more_1_liter' 'none' 'slight']
rectal_exam_feces         |    ['decreased' 'absent' 'normal' 'increased' 'serosanguious']
abdomen                   |    ['distend_small' 'distend_large' 'normal' 'firm' 'other']
abdomo_appearance         |    ['serosanguious' 'cloudy' 'clear']
surgical_lesion           |    ['yes' 'no']
```

```
cp_data                 |    ['no' 'yes']
temp_category           |    ['Normal' 'Hypothermia' 'Fever']
```

B) pain and mucous_membrane

Bernoulli Naive Bayes Classifier

Context:

Evaluating the performance of the Bernoulli Naive Bayes classifier not only through traditional classification metrics but also by analyzing the types of errors made can offer deeper insights into the model's behavior. A confusion matrix is particularly useful for this purpose as it visualizes the performance of an algorithm by showing the actual versus predicted classifications.

Task:

Generate a confusion matrix for the Bernoulli Naive Bayes model and analyze which outcome categories are most frequently misclassified.

Instructions:

1. **Prepare and Train the Model:**
   - Implement the Bernoulli Naive Bayes classifier and train it on the dataset.
   - Predict the outcomes using the test set.
2. **Generate a Confusion Matrix:**
   - Create a confusion matrix to visualize the model's predictions against the actual values.
3. **Analyze Misclassifications:**
   - Examine the confusion matrix to determine which category is most frequently misclassified as "Euthanized."

Question:

According to the confusion matrix generated from the Bernoulli Naive Bayes model, which category is most frequently incorrectly predicted as "Euthanized"?

Options:

A) Died

B) Lived

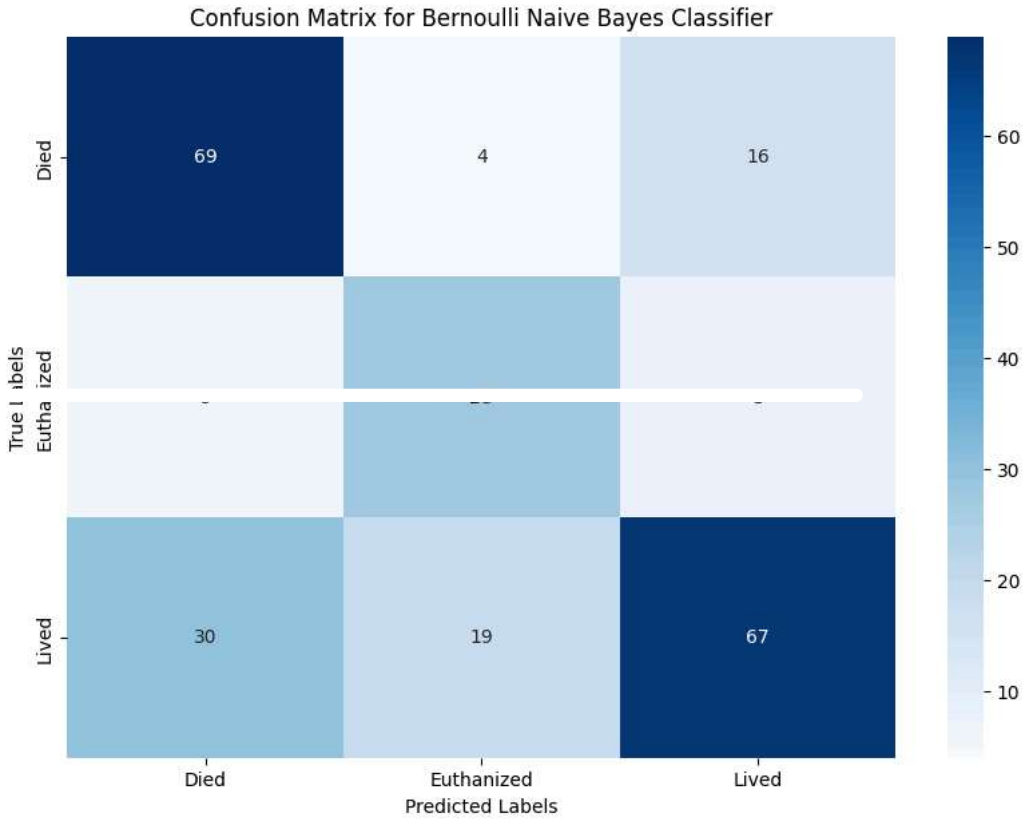
C) Both 'Died' and 'Lived' are equally misclassified as 'Euthanized'

D) Neither 'Died' nor 'Lived' is misclassified as 'Euthanized'

```
from sklearn.naive_bayes import BernoulliNB  # TODO: Correctly import the Bernoulli Naive Bayes classifier.
from sklearn.metrics import classification_report, confusion_matrix

# Encoding the 'outcome' variable
outcome_map = {'died': 0, 'euthanized': 1, 'lived': 2}
y_train_encoded = y_train.map(outcome_map)  # TODO: Apply the mapping to the 'y_train' series.
y_test_encoded = y_test.map(outcome_map)  # TODO: Apply the mapping to the 'y_test' series.

# Train Bernoulli Naive Bayes classifier
bernoulli_nb = BernoulliNB()
bernoulli_nb.fit(X_train, y_train_encoded)  # TODO: Fit the Bernoulli Naive Bayes model using training data.
y_pred_ber = bernoulli_nb.predict(X_test)  # TODO: Predict outcomes using the test set.

print("Bernoulli Naive Bayes Classification Report:")
print(classification_report(y_test_encoded, y_pred_ber, target_names=['Died', 'Euthanized', 'Lived']))  # TODO: Fill in the cl

# Generate the confusion matrix
cm = confusion_matrix(y_test_encoded, y_pred_ber)  # TODO: Provide the correct variables to generate the confusion matrix.
categories = ['Died', 'Euthanized', 'Lived']

# Plotting the confusion matrix
plt.figure(figsize=(10, 7))
sns.heatmap(cm, annot=True, fmt='g', cmap='Blues', xticklabels=categories, yticklabels=categories)
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
```

```
plt.title('Confusion Matrix for Bernoulli Naive Bayes Classifier')
plt.show()
```

```
Bernoulli Naive Bayes Classification Report:
              precision    recall  f1-score   support

        Died       0.66      0.78      0.71        89
  Euthanized       0.55      0.67      0.60        42
       Lived       0.74      0.58      0.65       116

    accuracy                           0.66       247
   macro avg       0.65      0.67      0.65       247
weighted avg       0.68      0.66      0.66       247
```



Assesments for Naive Bayes-II

Mucous Membrane Condition

Context:

The condition of a horse's mucous membrane can provide valuable insights into its circulatory and respiratory health, which are critical factors in determining the overall health status. Analyzing how different conditions of mucous membranes correlate with health outcomes can help in early diagnosis and management of health issues.

Task:

Investigate the relationship between the condition of the mucous membrane and health outcomes in horses.

Instructions:

1. **Group Data by Mucous Membrane Condition:** Use the dataset to group entries by 'mucous_membrane' and 'outcome' to analyze the distribution of health outcomes within each mucous membrane condition category.
2. **Visualize and Analyze Data:** Create a bar chart to visualize the count of each health outcome within the different mucous membrane condition categories. This will help identify any significant correlations between mucous membrane condition and health outcomes.

Question:

After analyzing the bar chart which depicts the count of health outcomes within each mucous membrane condition category, identify the correct statement regarding the correlation between mucous membrane condition and health outcomes.

Options:

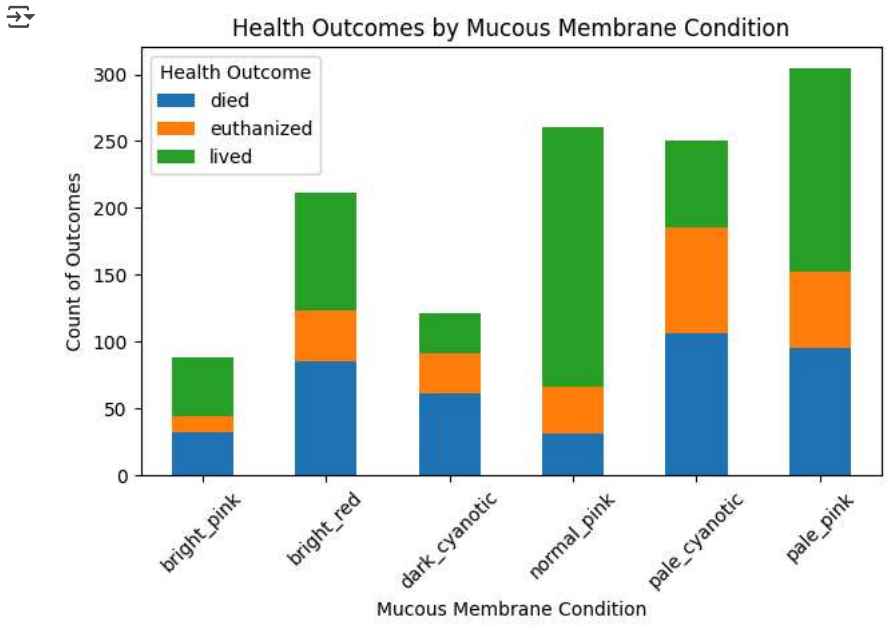
A) Horses with normal pink mucous membranes tend to have better health outcomes and higher survival rates.

B) A pale or cyanotic mucous membrane is strongly associated with a higher survival rate.

C) Horses with bright pink mucous membranes are more likely to be euthanized due to severe health issues.

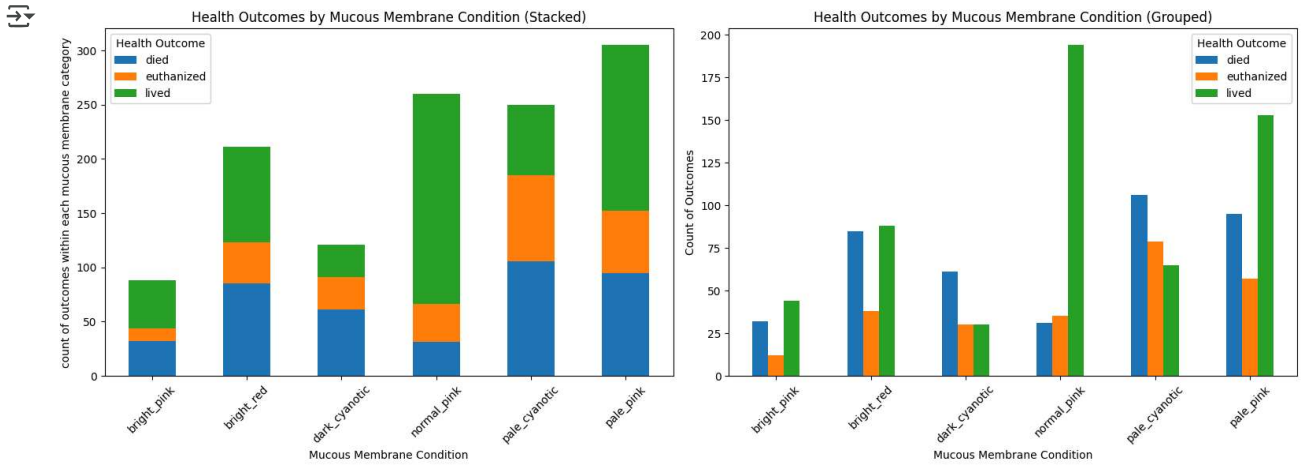
D) The color of the mucous membrane has no correlation with the health outcomes of horses.

```
# Grouping data by mucous membrane condition and outcome for plotting
grouped_data = df.groupby(["outcome","mucous_membrane"]).count()['id'].reset_index()  # TODO: Choose appropriate columns for g
grouped_data.columns = ['Mucous Membrane Condition', 'Outcome', 'Count']

# Create a pivot table for visualization
pivot_table = pd.pivot_table(grouped_data, values="Count", index="Outcome", columns="Mucous Membrane Condition", fill_value=0)

# Plotting the data
pivot_table.plot(kind='bar', stacked=True)
plt.title('Health Outcomes by Mucous Membrane Condition')
plt.xlabel('Mucous Membrane Condition')
plt.ylabel('Count of Outcomes')
plt.xticks(rotation=45)
plt.legend(title='Health Outcome')
plt.tight_layout()  # Adjust layout to not cut off labels

plt.show()
```




```
import matplotlib.pyplot as plt

cross_tab_a=pd.crosstab(df["mucous_membrane"],df["outcome"])
cross_tab_b=pd.crosstab(df["mucous_membrane"],df["outcome"])

# Create a figure with two subplots side by side
fig, axes = plt.subplots(1, 2, figsize=(16, 6)) # 1 row, 2 columns

# Plotting the data in grid (0,0) as stacked bar plot
cross_tab_a.plot(kind='bar', stacked=True, ax=axes[0]) # Stacked bar plot
axes[0].set_title("Health Outcomes by Mucous Membrane Condition (Stacked)") # Updated title
axes[0].set_xlabel("Mucous Membrane Condition")
axes[0].set_ylabel('count of outcomes within each mucous membrane category')
axes[0].tick_params(axis='x', rotation=45)
axes[0].legend(title='Health Outcome')
```

```
# Plotting the data in grid (0,1) as grouped bar plot
cross_tab_b.plot(kind='bar', stacked=False, ax=axes[1]) # Grouped bar plot
axes[1].set_title("Health Outcomes by Mucous Membrane Condition (Grouped)") # Updated title
axes[1].set_xlabel("Mucous Membrane Condition")
axes[1].set_ylabel('Count of Outcomes')
axes[1].tick_params(axis='x', rotation=45)
axes[1].legend(title='Health Outcome')

fig.tight_layout()  # Adjust layout to not cut off labels

plt.show()
```



- **A) Horses with normal pink mucous membranes tend to have better health outcomes and higher survival rates.**
- Pale pink mucous membranes is more common among Horses followed by normal pink.

Training and Evaluating the Gaussian Model

Context:

After preprocessing the dataset for the "Predict Health Outcomes of Horses" and training a Gaussian Naive Bayes model, the next step is to evaluate the classifier's performance. This involves encoding the target variable and analyzing the model's ability to predict each category.

Task:

Train and evaluate the Gaussian Naive Bayes classifier using the preprocessed features and encoded target. Use the classification report to identify which category the model has difficulty predicting.

Instructions:

1. **Encode the Target Variable:**
   - Map the 'outcome' categories to numeric values using the provided dictionary.
2. **Model Training:**
   - Train the Gaussian Naive Bayes classifier using the encoded training data.
3. **Model Evaluation:**
   - Evaluate the classifier's performance using the classification report to identify which category is most problematic.

Question:

Which category does the Gaussian Naive Bayes model struggle with the most, according to the classification report?

Options:

A) Died

B) Euthanized

C) Lived

D) All categories have nearly identical recall

```
from sklearn.naive_bayes import GaussianNB  # TODO: Import the Gaussian Naive Bayes classifier.
from sklearn.metrics import classification_report, confusion_matrix  # TODO: Import necessary functions for evaluation.

# Encoding the 'outcome' variable
outcome_map = {'died': 0, 'euthanized': 1, 'lived': 2}
y_train_encoded = y_train.map(outcome_map)  # TODO: Apply the mapping to the 'y_train' series.
y_test_encoded = y_test.map(outcome_map)  # TODO: Apply the mapping to the 'y_test' series.

# Initialize and train Gaussian Naive Bayes classifier
model = GaussianNB()
model.fit(X_train, y_train_encoded)  # TODO: Fit the model using the training data.

# Predict on the test set
y_pred = model.predict(X_test)  # TODO: Make predictions using the test data.

# Generate the classification report
report = classification_report(y_test_encoded, y_pred, target_names=['Died', 'Euthanized', 'Lived'])  # TODO: Provide the corr
print(report)
```

```
              precision    recall  f1-score   support

        Died       0.49      0.98      0.65        89
  Euthanized       0.56      0.57      0.56        42
       Lived       0.88      0.20      0.32       116

    accuracy                           0.54       247
   macro avg       0.64      0.58      0.51       247
weighted avg       0.69      0.54      0.48       247
```

C) Lived

Cross-Validation with Gaussian

Context:

Cross-validation is a robust technique for assessing the generalizability of a predictive model beyond the specific data used during training. Using 5-fold cross-validation with the Gaussian Naive Bayes classifier provides insight into the model's performance across different subsets of the dataset.

Task:

Apply 5-fold cross-validation using the Gaussian Naive Bayes Classifier on the "Predict Health Outcomes of Horses" dataset, utilizing the 'f1_weighted' scoring metric to evaluate the model's performance.

Instructions:

1. **Set Up Cross-Validation:**
    - Use the Gaussian Naive Bayes classifier.
    - Implement 5-fold cross-validation.
    - Apply the 'f1_weighted' metric to measure model performance across folds.
2. **Calculate the Average Score:**
    - Determine the average 'f1_weighted' score across all cross-validation folds to assess the overall effectiveness of the model.

Question:

1. What is the average 'f1_weighted' score value using 5-fold cross-validation with the Gaussian Naive Bayes Classifier?
2. Is this 'f1_weighted' score higher or lower than the F1 score from the previously generated classification report for the category the model struggled with the most?

Options:

A) Mean F1 score: 0.5, Higher

B) Mean F1 score: 0.6, Higher

C) Mean F1 score: 0.7, Lower

D) Mean F1 score: 0.8, Lower

```
from sklearn.model_selection import cross_val_score  # TODO: Import the function to perform cross-validation.

# Perform 5-fold cross-validation using 'f1_weighted' as the scoring metric
# Review the scoring parameters available in the documentation to ensure correct usage. https://scikit-learn.org/stable/module
cv_scores = cross_val_score(model, X_train, y_train_encoded, cv=5, scoring='f1_weighted')  # TODO: Initialize the classifier,

# Calculate the average of the cross-validation scores
average_cv_score = cv_scores.mean()  # TODO: Determine the appropriate method to calculate the mean of the scores.
print(f"Average F1 weighted score across 5 folds: {average_cv_score:.2f}")
```

```
Average F1 weighted score across 5 folds: 0.50
```

Multinomial Naive Bayes Evaluation

Context:

After training a Gaussian Naive Bayes model on the "Predict Health Outcomes of Horses" dataset and obtaining its performance metrics, the next step involves evaluating a Multinomial Naive Bayes model. This evaluation will use the same dataset, but the continuous features will be scaled to `non-negative` values suitable for Multinomial Naive Bayes.

Task:

Evaluate the performance of the Multinomial Naive Bayes classifier after scaling the features to ensure all are non-negative, as required by the model.

Instructions:

1. **Data Preparation:**
   - Scale the continuous features using `MinMaxScaler` to ensure all feature values are non-negative.
2. **Model Training and Evaluation:**
   - Train a Multinomial Naive Bayes classifier on the scaled data.
   - Generate a classification report to assess the model's performance across different outcome categories.

Question:

Based on the classification report for the Multinomial Naive Bayes model, which category has the highest recall, indicating the best performance in correctly identifying true positive cases?

Options:

A) Died

B) Euthanized

C) Lived

D) All categories have nearly identical recall

```
from sklearn.naive_bayes import MultinomialNB  # TODO: Correctly import the Multinomial Naive Bayes classifier.
from sklearn.preprocessing import MinMaxScaler  # TODO: Correctly import the MinMaxScaler for data normalization.
from sklearn.metrics import classification_report, confusion_matrix # Import classification_report and confusion_matrix

# Initialize MinMaxScaler
scaler = MinMaxScaler()  # TODO: Instantiate the scaler.

# Scale X_train and X_test
X_train_scaled = scaler.fit_transform(X_train)  # TODO: Fit and Apply the scaler to train data.
X_test_scaled = scaler.transform(X_test)  # TODO: Apply the scaler to test data.

# Train Multinomial Naive Bayes classifier
multinomial_nb = MultinomialNB()
multinomial_nb.fit(X_train_scaled, y_train_encoded)  # TODO: Fit the Multinomial Naive Bayes model to the scaled training data
y_pred_multinomial = multinomial_nb.predict(X_test_scaled)  # TODO: Predict the test set results with the trained model.
```

```
print("Multinomial Naive Bayes Classification Report:")
print(classification_report(y_test_encoded, y_pred_multinomial, target_names=['Died', 'Euthanized', 'Lived']))  # TODO: Provid
```

```
Multinomial Naive Bayes Classification Report:
              precision    recall  f1-score   support

        Died       0.67      0.56      0.61        89
  Euthanized       0.62      0.57      0.59        42
       Lived       0.62      0.72      0.67       116

    accuracy                           0.64       247
   macro avg       0.64      0.62      0.62       247
weighted avg       0.64      0.64      0.63       247
```

Optimizing Laplace Smoothing for Gaussian Naive Bayes Classifier

Context:

Laplace smoothing is a technique used to handle the issue of zero probability in Naive Bayes classifiers. While typically more relevant for discrete data in Multinomial and Bernoulli Naive Bayes, experimenting with this parameter in Gaussian Naive Bayes can provide insights into its effect on model robustness, particularly when handling datasets with sparse features or small sample sizes.

Task:

Determine the best Laplace smoothing parameter (alpha) for a Gaussian Naive Bayes classifier applied to the "Predict Health Outcomes of Horses" dataset. This involves tuning the alpha parameter to improve model performance, especially in handling data with potential zero-frequency issues.

Instructions:

1. **Setup Parameter Tuning:**
   - Use a range of alpha values to determine which provides the best balance between bias and variance.
   - Apply cross-validation to evaluate each model configuration under different alpha settings.
2. **Evaluate Model Performance:**
   - Measure the performance 'f1_weighted' of the Gaussian Naive Bayes classifier for each alpha setting.
   - Identify the alpha value that results in the highest average performance across cross-validation folds.
3. **Implement and Discuss Results:**
   - Implement the model using the optimal alpha.
   - Discuss how Laplace smoothing impacts the performance of a Gaussian Naive Bayes classifier.

Question:

After tuning the Laplace smoothing parameter for the Gaussian Naive Bayes model on the "Predict Health Outcomes of Horses" dataset, which alpha value resulted in the highest f1_weighted score?

Options:

A) Alpha = 0.0, Accuracy = 65%

B) Alpha = 0.8, Accuracy = 64%

C) Alpha = 0.5, Accuracy = 75%

D) Alpha = 0.3, Accuracy = 53%

```
from sklearn.model_selection import GridSearchCV  # TODO: Import the necessary class for optimizing hyperparameters.
import numpy as np

# Define a range of alpha values to test
alpha_range = np.linspace(0.0, 1.0, 11)

# Setup GridSearchCV to find the best alpha
param_grid = {'var_smoothing': alpha_range}  # TODO: Specify the parameter name for Laplace smoothing in GaussianNB.
gnb = GaussianNB()
clf = GridSearchCV(gnb, param_grid, cv=5, scoring='f1_weighted')  # TODO: Fill in the GridSearchCV constructor with the correc
clf.fit(X_train, y_train)  # TODO: Fit the GridSearchCV to the training data.

# Best alpha and its score
best_alpha = clf.best_params_['var_smoothing']  # TODO: Extract the best alpha value from clf.
```

```
best_score = clf.best_score_  # TODO: Extract the best score from clf.
print(f"Best alpha: {best_alpha}")
print(f"Best scoring f1_weighted: {best_score:.2f}")
```

```
Best alpha: 0.8
Best scoring f1_weighted: 0.64
```

Match the Following attributes

Context:

Understanding how different Naive Bayes classifiers calculate feature importance is crucial for interpreting model outcomes and for feature selection. Each classifier type has specific methods or attributes that help in identifying the contribution of each feature towards making a prediction.

Task:

Match the following Naive Bayes models to the correct method or attribute used for determining feature importance, which provides insights such as $(P(x_i \mid y))$ or the mean of each feature per class. Use the scikit-learn documentation to find the parameters that return these values.

Instructions:

1. **Research**: Look into the scikit-learn documentation for Naive Bayes classifiers to identify the specific attributes associated with each model.
2. **Match**: Connect each model listed on the left with its corresponding method or attribute on the right that is used to determine feature importance.

Match the Following Characteristics

Models	Characteristics
A. Gaussian Naive Bayes	I. feature_log_prob_
B. Multinomial Naive Bayes	II. feature_importances_
C. Bernoulli Naive Bayes	III. epsilon_
	IV: theta_

Options:

- A) A-I, B-III, C-II
- B) A-IV, B-I, C-I
- C) A-IV, B-I, C-II
- D) A-I, B-IV, C-I

B) A-IV, B-I, C-I

Evaluating Feature Importance in Gaussian Naive Bayes Model

Context:

Understanding which features most influence the predictions of a Gaussian Naive Bayes classifier can enhance interpretability and guide further model refinement. This model evaluates feature importance based on the mean values of features conditioned on each class, reflecting the significance of each feature in class discrimination.

Task:

Analyze feature importance in a Gaussian Naive Bayes model trained on a dataset, using the mean of each feature across classes to determine their impact on the model's predictions.

Instructions:

1. **Model Setup and Training:**
   - Initialize the Gaussian Naive Bayes model with `var_smoothing` set to 0.3 to stabilize the calculation by adjusting the variance of each feature.
   - Train the model using the training sets `X_train` and `y_train`.

2. **Feature Importance Calculation:**
    - Extract the mean of each feature for each class from the model. These means are critical as Gaussian Naive Bayes assumes features are normally distributed, and the mean of these distributions plays a significant role in class separation.
    - Calculate the `absolute` values of these means to focus on the magnitude of feature values irrespective of their direction.
3. **Summarize Feature Importance:**
    - Aggregate these mean values across all classes to get a single importance score per feature.
    - Rank these features based on their importance scores to identify the most influential features.
4. **Visualization:**
    - Plot these importance scores using a horizontal bar chart to visually compare the significance of each feature.

Question:

After implementing the above steps, which features were identified as the most important based on the mean absolute values of the class conditional distributions?

Options:

A) rectal_temp, age, peristalsis

B) lesion_2, pain, cp_data

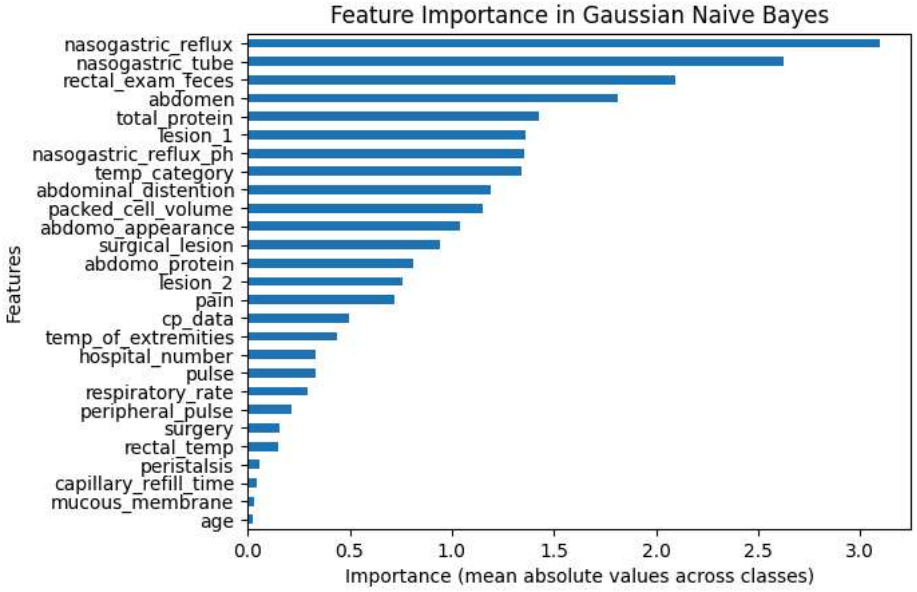
C) nasogastric_reflux, nasogastric_tube, rectal_exam_feces

D) total_protein, lesion_1, temp_category

```
# Train the Gaussian Naive Bayes model
model = GaussianNB(var_smoothing  = 0.3) # TODO: Set the smoothing parameter to 0.3.
model.fit(X_train, y_train)

# TODO: Use the appropriate numpy function to calculate the absolute values of the means
feature_importances = np.mean(np.abs(model.theta_))   # TODO: Access the attribute that stores the class conditional means.

# Create a DataFrame to view these importances
importance_df = pd.DataFrame(feature_importances, columns=X.columns)

# Summing the importances across classes to get an overall importance score for each feature
importance_df.loc['Mean Importance'] = importance_df.mean(axis=0)  # TODO: Aggregate the means to get a single importance scor
sorted_importance = importance_df.loc['Mean Importance'].sort_values(ascending=False)   # TODO: Sort the features based on imp

print("Feature Importance based on absolute means of class conditional distributions:")
print(sorted_importance)

# Optionally, visualize these importances
import matplotlib.pyplot as plt
sorted_importance.plot(kind='barh')
plt.title('Feature Importance in Gaussian Naive Bayes')
plt.ylabel('Features')
plt.xlabel('Importance (mean absolute values across classes)')
plt.show()


import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.naive_bayes import GaussianNB

# Train the Gaussian Naive Bayes model
# Use the best var_smoothing found in the previous step if available, otherwise use 0.3 as specified in the markdown
model = GaussianNB(var_smoothing=0.3)
model.fit(X_train, y_train_encoded) # Fit with encoded y_train

# Calculate the absolute values of the class conditional means
# The class conditional means are stored in the theta_ attribute
abs_means = np.abs(model.theta_)

# Calculate the mean of the absolute values across classes for each feature
# axis=0 calculates the mean down the columns (across the classes)
feature_importances = np.mean(abs_means, axis=0)

# Create a DataFrame to view these importances
# The columns of this DataFrame should correspond to the feature names before transformation
importance_df = pd.DataFrame({'Feature': X.columns, 'Importance': feature_importances})
```

```
# Sort the features based on importance
sorted_importance = importance_df.sort_values(by='Importance', ascending=False)

print("Feature Importance based on absolute means of class conditional distributions:")
print(sorted_importance)

# Optionally, visualize these importances

sorted_importance.plot(kind='barh', x='Feature', y='Importance', legend=False)
plt.title('Feature Importance in Gaussian Naive Bayes')
plt.ylabel('Features')
plt.xlabel('Importance (mean absolute values across classes)')
plt.gca().invert_yaxis() # Invert y-axis to show most important features at the top
plt.show()
```

```
Feature Importance based on absolute means of class conditional distributions:
                   Feature  Importance
14     nasogastric_reflux    3.093584
13       nasogastric_tube    2.627840
16      rectal_exam_feces    2.094395
17                abdomen    1.809921
19          total_protein    1.424576
23                lesion_1    1.359208
15  nasogastric_reflux_ph    1.353643
26          temp_category    1.339915
12   abdominal_distention    1.192330
18     packed_cell_volume    1.148624
20      abdomo_appearance    1.042088
22        surgical_lesion    0.939177
21         abdomo_protein    0.811958
24                lesion_2    0.756599
10                    pain    0.716326
25                cp_data    0.496992
6     temp_of_extremities    0.440712
2         hospital_number    0.333712
4                   pulse    0.332115
5        respiratory_rate    0.291780
7        peripheral_pulse    0.215117
0                  surgery    0.153676
3             rectal_temp    0.151303
11            peristalsis    0.055472
9   capillary_refill_time    0.047601
8         mucous_membrane    0.030234
1                     age    0.028012
```



Start coding or generate with AI.