Analysing biomarkers to determine the presence of liver cirrhosis.

ASDS 6302 final project

Utkarsh Pant

The aim of this project is to develop Logistic Regression and Support Vector Machine models that can predict the presence of liver cirrhosis on the basis of certain biomarkers. **The original research that produced the dataset also sought to investigate the gender-based disparity in the study of the disease, leading to marked difference in patient treatment and outcomes. Hence, the "imbalance" between data available for male and female patients is a key feature of the dataset.**

In [ ]:
```
import pandas
import numpy

import seaborn
import matplotlib.pyplot as plot

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC

from sklearn.metrics import accuracy_score, confusion_matrix, roc_curve, roc_auc_score
```

In [ ]:
```
# Load the dataset and assign column names
liver_data = pandas.read_csv("./dataset/ilpd.csv")
liver_data.columns = ["Age",
                      "Gender",
                      "TB",
                      "DB",
                      "Alkphos",
                      "Sgpt",
                      "Sgot",
                      "TP",
                      "ALB",
                      "AGRatio",
                      "Target"]
liver_data.head()
```

Out[ ]:

	Age	Gender	TB	DB	Alkphos	Sgpt	Sgot	TP	ALB	AGRatio	Target
0	62	Male	10.9	5.5	699	64	100	7.5	3.2	0.74	1
1	62	Male	7.3	4.1	490	60	68	7.0	3.3	0.89	1
2	58	Male	1.0	0.4	182	14	20	6.8	3.4	1.00	1
3	72	Male	3.9	2.0	195	27	59	7.3	2.4	0.40	1
4	46	Male	1.8	0.7	208	19	14	7.6	4.4	1.30	1

Descriptive statistics and data overview

Checking for any NA or otherwise missing values in the dataset:

In [ ]:
```
liver_data.isna().sum()
```

Out[ ]:
```
Age         0
Gender      0
TB          0
DB          0
Alkphos     0
Sgpt        0
Sgot        0
TP          0
ALB         0
AGRatio     4
Target      0
dtype: int64
```

There appear to be no missing values in the dataset, except for the `AGRatio` column. Upon inspection, we see:

In [ ]:
```
liver_data.loc[pandas.isna(liver_data["AGRatio"]), ]
```

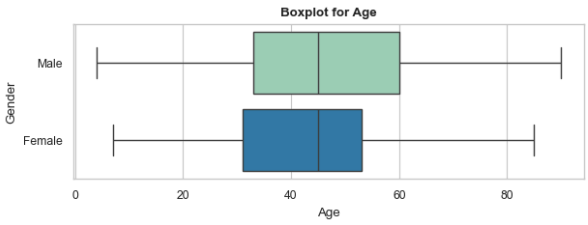
Out[ ]:

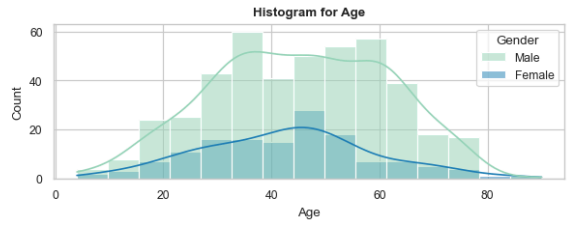
	Age	Gender	TB	DB	Alkphos	Sgpt	Sgot	TP	ALB	AGRatio	Target
208	45	Female	0.9	0.3	189	23	33	6.6	3.9	NaN	1
240	51	Male	0.8	0.2	230	24	46	6.5	3.1	NaN	1
252	35	Female	0.6	0.2	180	12	15	5.2	2.7	NaN	2
311	27	Male	1.3	0.6	106	25	54	8.5	4.8	NaN	2

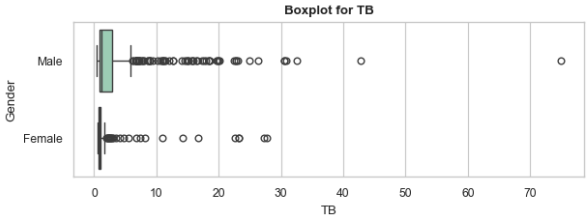
In [ ]:
```
seaborn.set_theme("paper")
seaborn.set_style("whitegrid")
```

In [ ]:
```
exclude = ["Gender", "Target"]
continuous = [col for col in liver_data.columns if col not in exclude]

figure, axes = plot.subplots(len(continuous), 2, figsize=(15, 3 * len(continuous)))
axes = axes.flatten()

for i, axes in enumerate(axes):
    # Plot boxplot
    if i % 2 == 0:
        seaborn.boxplot(x=liver_data[continuous[i // 2]], ax=axes, palette = 'YlGnBu', y="Gender"
        axes.set_title(f'Boxplot for {continuous[i // 2]}', fontdict={'fontweight': 'bold'})
    # # Plot histogram
    else:
        seaborn.histplot(x=liver_data[continuous[i // 2]], ax=axes, kde=True, hue="Gender", palet
        axes.set_title(f'Histogram for {continuous[i // 2]}', fontdict={'fontweight': 'bold'})

plot.subplots_adjust(hspace=0.5)  # Adjust the spacing between rows
```

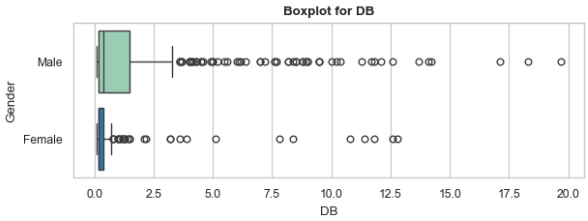
```
C:\Users\utkar\AppData\Local\Temp\ipykernel_20640\2037598917.py:10: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the
`y` variable to `hue` and set `legend=False` for the same effect.

  seaborn.boxplot(x=liver_data[continuous[i // 2]], ax=axes, palette = 'YlGnBu', y="Gender", data=
liver_data)
C:\Users\utkar\AppData\Local\Temp\ipykernel_20640\2037598917.py:10: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the
`y` variable to `hue` and set `legend=False` for the same effect.

  seaborn.boxplot(x=liver_data[continuous[i // 2]], ax=axes, palette = 'YlGnBu', y="Gender", data=
liver_data)
C:\Users\utkar\AppData\Local\Temp\ipykernel_20640\2037598917.py:10: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the
`y` variable to `hue` and set `legend=False` for the same effect.

  seaborn.boxplot(x=liver_data[continuous[i // 2]], ax=axes, palette = 'YlGnBu', y="Gender", data=
liver_data)
C:\Users\utkar\AppData\Local\Temp\ipykernel_20640\2037598917.py:10: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the
`y` variable to `hue` and set `legend=False` for the same effect.

  seaborn.boxplot(x=liver_data[continuous[i // 2]], ax=axes, palette = 'YlGnBu', y="Gender", data=
liver_data)
C:\Users\utkar\AppData\Local\Temp\ipykernel_20640\2037598917.py:10: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the
`y` variable to `hue` and set `legend=False` for the same effect.

  seaborn.boxplot(x=liver_data[continuous[i // 2]], ax=axes, palette = 'YlGnBu', y="Gender", data=
liver_data)
C:\Users\utkar\AppData\Local\Temp\ipykernel_20640\2037598917.py:10: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the
`y` variable to `hue` and set `legend=False` for the same effect.

  seaborn.boxplot(x=liver_data[continuous[i // 2]], ax=axes, palette = 'YlGnBu', y="Gender", data=
liver_data)
C:\Users\utkar\AppData\Local\Temp\ipykernel_20640\2037598917.py:10: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the
`y` variable to `hue` and set `legend=False` for the same effect.

  seaborn.boxplot(x=liver_data[continuous[i // 2]], ax=axes, palette = 'YlGnBu', y="Gender", data=
liver_data)
C:\Users\utkar\AppData\Local\Temp\ipykernel_20640\2037598917.py:10: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the
`y` variable to `hue` and set `legend=False` for the same effect.

  seaborn.boxplot(x=liver_data[continuous[i // 2]], ax=axes, palette = 'YlGnBu', y="Gender", data=
liver_data)
C:\Users\utkar\AppData\Local\Temp\ipykernel_20640\2037598917.py:10: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the
`y` variable to `hue` and set `legend=False` for the same effect.

  seaborn.boxplot(x=liver_data[continuous[i // 2]], ax=axes, palette = 'YlGnBu', y="Gender", data=
liver_data)
```

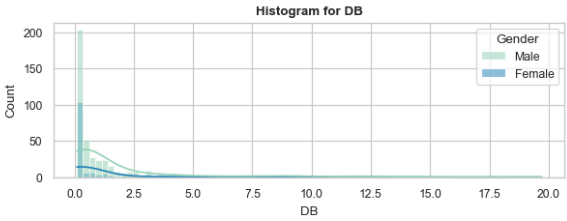

Boxplot for Age
Gender
Male
Female
0
20
40
60
80
Age

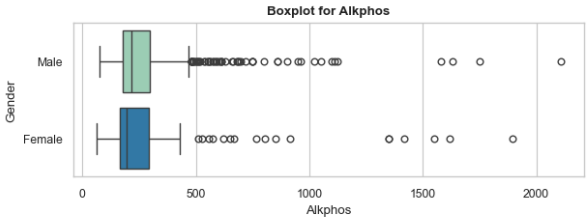

Histogram for Age
Count
0
20
40
60
0
20
40
60
80
Age
Gender
Male
Female

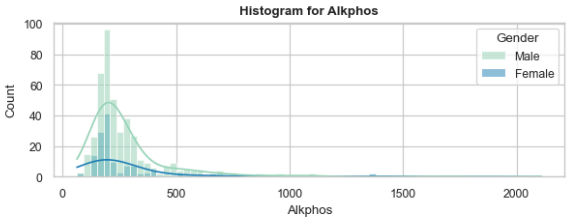

Boxplot for TB
Gender
Male
Female
0
10
20
30
40
50
60
70
TB

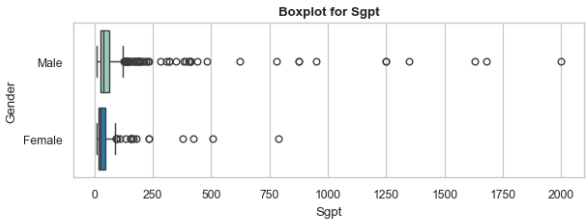

Histogram for TB
Count
0
50
100
150
0
10
20
30
40
50
60
70
TB
Gender
Male
Female

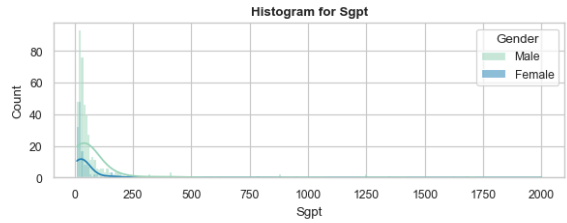

Boxplot for DB
Gender
Male
Female
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
DB

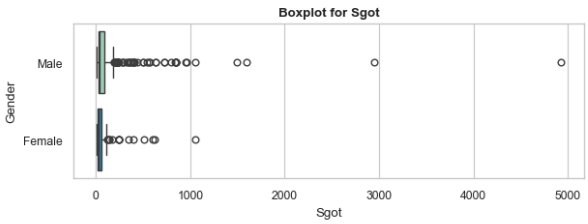

Histogram for DB
Count
0
50
100
150
200
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
DB
Gender
Male
Female

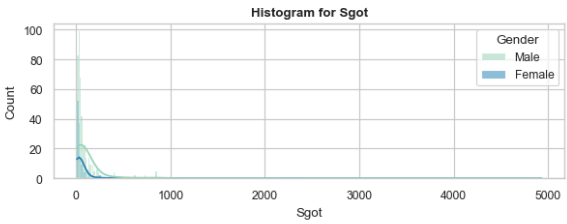

Boxplot for Alkphos
Gender
Male
Female
0
500
1000
1500
2000
Alkphos

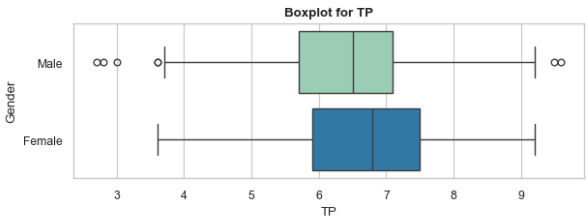

Histogram for Alkphos
Count
0
20
40
60
80
100
0
500
1000
1500
2000
Alkphos
Gender
Male
Female

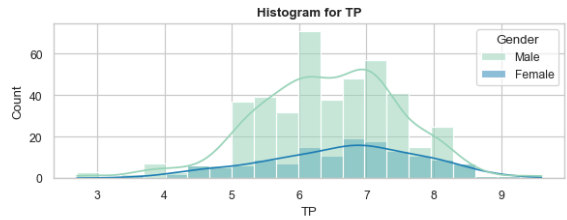

Boxplot for Sgpt
Gender
Male
Female
0
250
500
750
1000
1250
1500
1750
2000
Sgpt

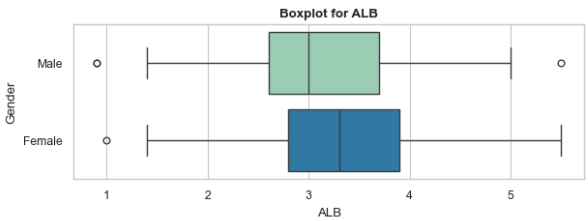

Histogram for Sgpt
Count
0
20
40
60
80
0
250
500
750
1000
1250
1500
1750
2000
Sgpt
Gender
Male
Female

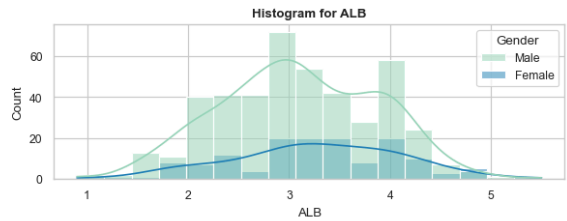

Boxplot for Sgot
Gender
Male
Female
0
1000
2000
3000
4000
5000
Sgot

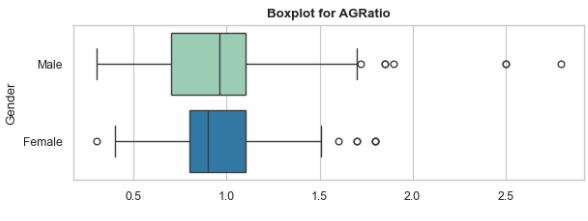

Histogram for Sgot
Count
0
20
40
60
80
100
0
1000
2000
3000
4000
5000
Sgot
Gender
Male
Female

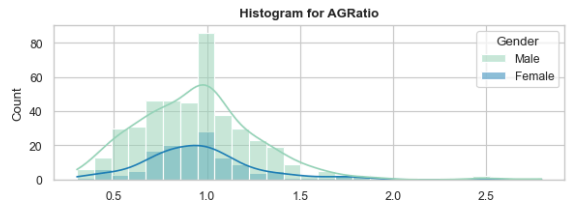

Boxplot for TP
Gender
Male
Female
3
4
5
6
7
8
9
TP


Histogram for TP
Count
0
20
40
60
3
4
5
6
7
8
9
TP
Gender
Male
Female


Boxplot for ALB
Gender
Male
Female
1
2
3
4
5
ALB


Histogram for ALB
Count
0
20
40
60
1
2
3
4
5
ALB
Gender
Male
Female

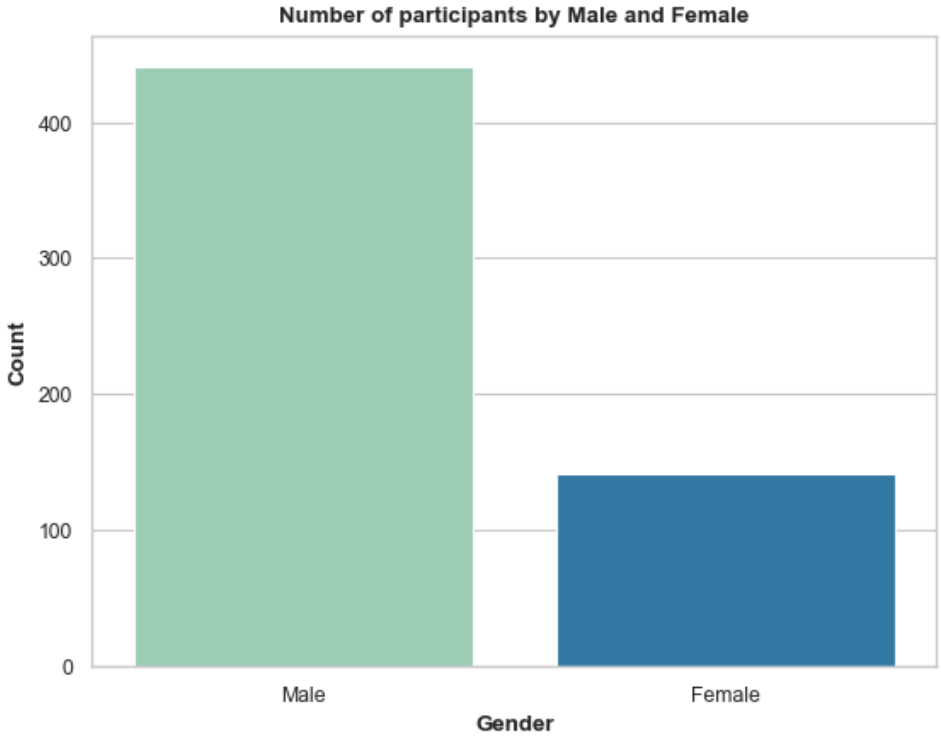

Boxplot for AGRatio
Gender
Male
Female
0.5
1.0
1.5
2.0
2.5


Histogram for AGRatio
Count
0
20
40
60
80
0.5
1.0
1.5
2.0
2.5
Gender
Male
Female

In [ ]:
```
gender_countplot = seaborn.countplot(data = liver_data, x = "Gender", palette = "YlGnBu")
gender_countplot.set_title("Number of participants by Male and Female", fontdict={"fontweight": "
gender_countplot.set_ylabel("Count", fontdict={"fontweight": "bold"})
gender_countplot.set_xlabel("Gender", fontdict={"fontweight": "bold"})
```

```
C:\Users\utkar\AppData\Local\Temp\ipykernel_20640\3016109195.py:1: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the
`x` variable to `hue` and set `legend=False` for the same effect.

  gender_countplot = seaborn.countplot(data = liver_data, x = "Gender", palette = "YlGnBu")
```

Out[ ]:
```
Text(0.5, 0, 'Gender')
```

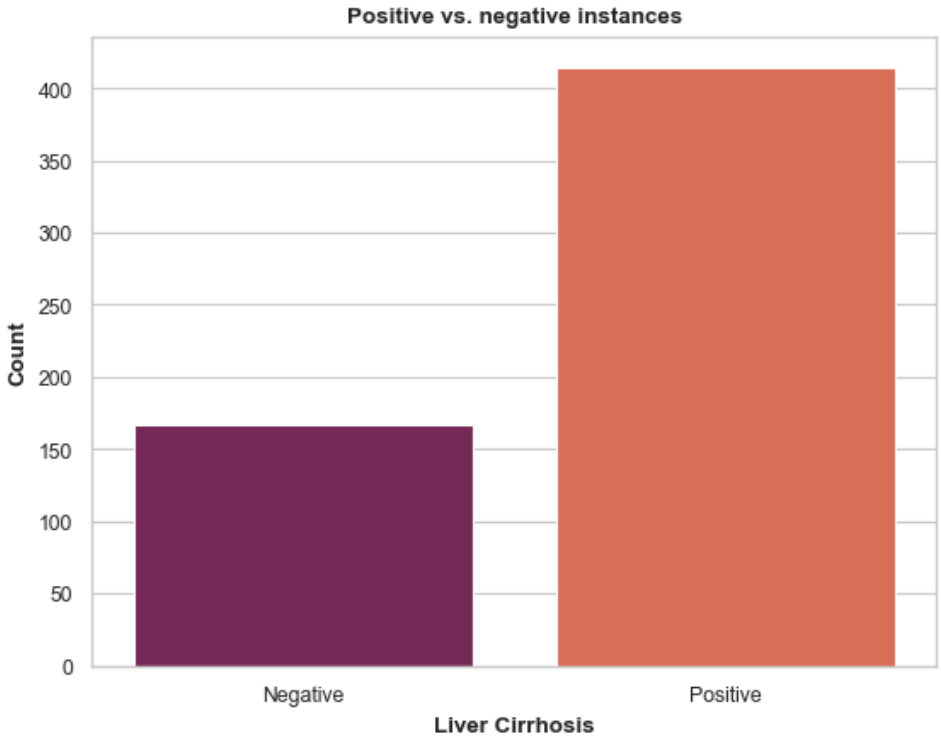


In [ ]:
```
liver_data.loc[liver_data['Target'] == 2, 'Target'] = 0 # healthy
liver_data.loc[liver_data['Target'] == 1, 'Target'] = 1
```

In [ ]:
```
target_countplot = seaborn.countplot(data = liver_data, x = "Target", palette = "rocket")
target_countplot.set_title("Positive vs. negative instances", fontdict={"fontweight": "bold"})
target_countplot.set_ylabel("Count", fontdict={"fontweight": "bold"})
target_countplot.set_xlabel("Liver Cirrhosis", fontdict={"fontweight": "bold"})
target_countplot.set_xticklabels(["Negative", "Positive"])
```

```
C:\Users\utkar\AppData\Local\Temp\ipykernel_20640\158615405.py:1: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the
`x` variable to `hue` and set `legend=False` for the same effect.

  target_countplot = seaborn.countplot(data = liver_data, x = "Target", palette = "rocket")
C:\Users\utkar\AppData\Local\Temp\ipykernel_20640\158615405.py:5: UserWarning: set_ticklabels() sh
ould only be used with a fixed number of ticks, i.e. after set_ticks() or using a FixedLocator.
  target_countplot.set_xticklabels(["Negative", "Positive"])
```

Out[ ]:
```
[Text(0, 0, 'Negative'), Text(1, 0, 'Positive')]
```

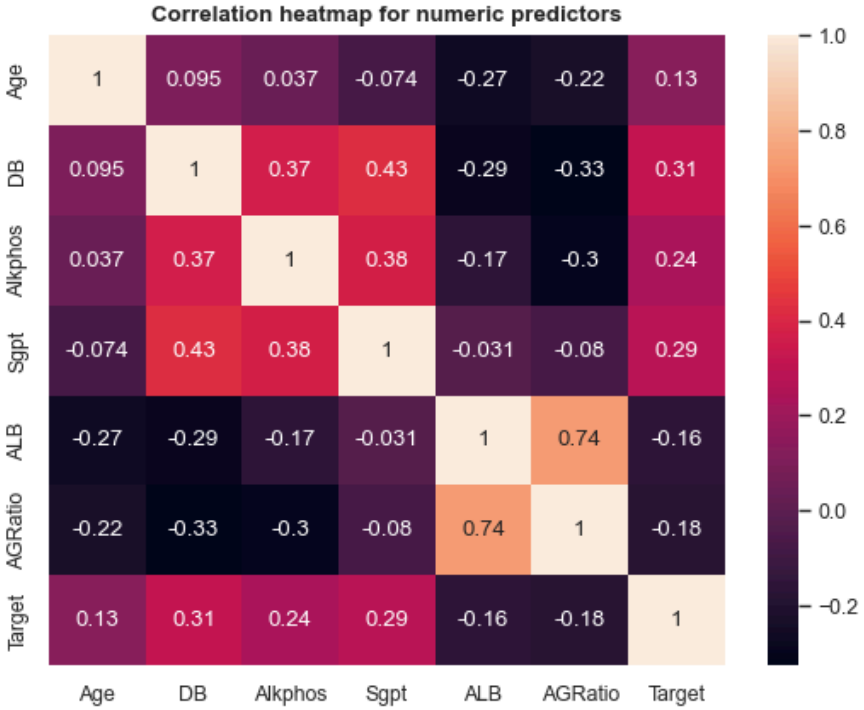


In this case, even though *significant* outliers exist in almost all features, we will not remove them as they are likely to be genuine data points. Imputing or removing outliers tends to "skew" data towards an "average" representation, which may dilute the disparity between sexes that we are trying to capture.

Correlation analysis + picking significant predictors

In [ ]:
```
import matplotlib.pyplot as plot

correlation = liver_data.corr(numeric_only=True)
heatmap = seaborn.heatmap(correlation, annot=True, cmap="rocket")
heatmap.set_title("Correlation heatmap for numeric predictors", fontdict={"fontweight": "bold"})
```

Out[ ]:
```
Text(0.5, 1.0, 'Correlation heatmap for numeric predictors')
```



Between the features, we see that `TB` and `DB` are highly correlated, as are `TP` and `ALB`. This is expected, as the former pair are both measures of bilirubin levels, while the latter pair are measures of protein levels. We must drop *one* feature in either pair from our model, in order to reduce redundant information. Hence, we drop `TB` and `TP` from the respective pairs since they are less correlated with `Target` as their counterparts. Similarly, between `Sgpt` and `Sgot`, we drop `Sgot` as it is less correlated with `Target`.

In [ ]:
```
liver_data.drop(['TB', 'TP', 'Sgot'], axis = 1, inplace = True)
liver_data.head()
```

Out[ ]:

	Age	Gender	DB	Alkphos	Sgpt	ALB	AGRatio	Target
0	62	Male	5.5	699	64	3.2	0.74	1
1	62	Male	4.1	490	60	3.3	0.89	1
2	58	Male	0.4	182	14	3.4	1.00	1
3	72	Male	2.0	195	27	2.4	0.40	1
4	46	Male	0.7	208	19	4.4	1.30	1

Data preprocessing

All features are continuous, and have no missing values. Hence, very little pre-processing is required. The only preprocessing needed is the label encoding of the `Gender` feature, which has the labels, "Male" and "Female".

In [ ]:
```
# encode male as 1 and female as 0
liver_data.loc[liver_data["Gender"] == "Male", "Gender"] = 1
liver_data.loc[liver_data["Gender"] == "Female", "Gender"] = 0
```

In [ ]:
```
# validating our encoding was successful
liver_data["Gender"].value_counts()
```

Out[ ]:
```
Gender
1    441
0    141
Name: count, dtype: int64
```

Since 2 observations are *Male* and 2 are *Female* participants, we can safely drop these observations from the dataset for simplicity.

In [ ]:
```
liver_data.dropna(subset=["AGRatio"], inplace=True)
```

We can also impute the outliers in our features, so as to base our modelling and analysis *only* on the underlying distributions for each continuous variable. Due to the lack of subject-matter expertise and the simple scope, we cannot determine the "true" values of these outliers, and hence we will impute them with the upper and lower "thresholds".

In [ ]:
```
# imputing outliers

for column in continuous:
    Q1 = numpy.quantile(liver_data[column], 0.25)
    Q3 = numpy.quantile(liver_data[column], 0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    liver_data.loc[liver_data[column] < lower_bound, column] = lower_bound
    liver_data.loc[liver_data[column] > upper_bound, column] = upper_bound
```

```
C:\Users\utkar\AppData\Local\Temp\ipykernel_20640\36010521.py:9: FutureWarning: Setting an item of incompatible dtype is deprecated and will raise an error in a future version of pandas. Value '-8.875' has dtype incompatible with int64, please explicitly cast to a compatible dtype first.
  liver_data.loc[liver_data[column] < lower_bound, column] = lower_bound
C:\Users\utkar\AppData\Local\Temp\ipykernel_20640\36010521.py:9: FutureWarning: Setting an item of incompatible dtype is deprecated and will raise an error in a future version of pandas. Value '-33.375' has dtype incompatible with int64, please explicitly cast to a compatible dtype first.
  liver_data.loc[liver_data[column] < lower_bound, column] = lower_bound
```

Scaling the data

In [ ]:
```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
liver_data[continuous] = scaler.fit_transform(liver_data[continuous])

liver_data.head()
```

Out[ ]:

	Age	Gender	DB	Alkphos	Sgpt	ALB	AGRatio	Target
0	1.064979	1	2.073315	2.098953	0.470077	0.077750	-0.676143	1
1	1.064979	1	2.073315	2.098953	0.350730	0.203630	-0.171698	1
2	0.818065	1	-0.486781	-0.637668	-1.021760	0.329510	0.198229	1
3	1.682266	1	1.119554	-0.519130	-0.633882	-0.929292	-1.819551	1
4	0.077321	1	-0.185594	-0.400593	-0.872576	1.588312	1.207119	1

Training and testing the models

Splitting the data into training and testing sets

In [ ]:
```
X = liver_data.drop("Target", axis = 1)
y = liver_data["Target"]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state = 8093)
```

Logistic Regression

In [ ]:
```
logisticRegression = LogisticRegression()
logisticRegression.fit(X_train, y_train)

y_pred_lr = logisticRegression.predict(X_test)
```

Accuracy

In [ ]:
```
accuracy_lr = accuracy_score(y_test, y_pred_lr)
accuracy_lr
```

Out[ ]:
```
0.6954022988505747
```

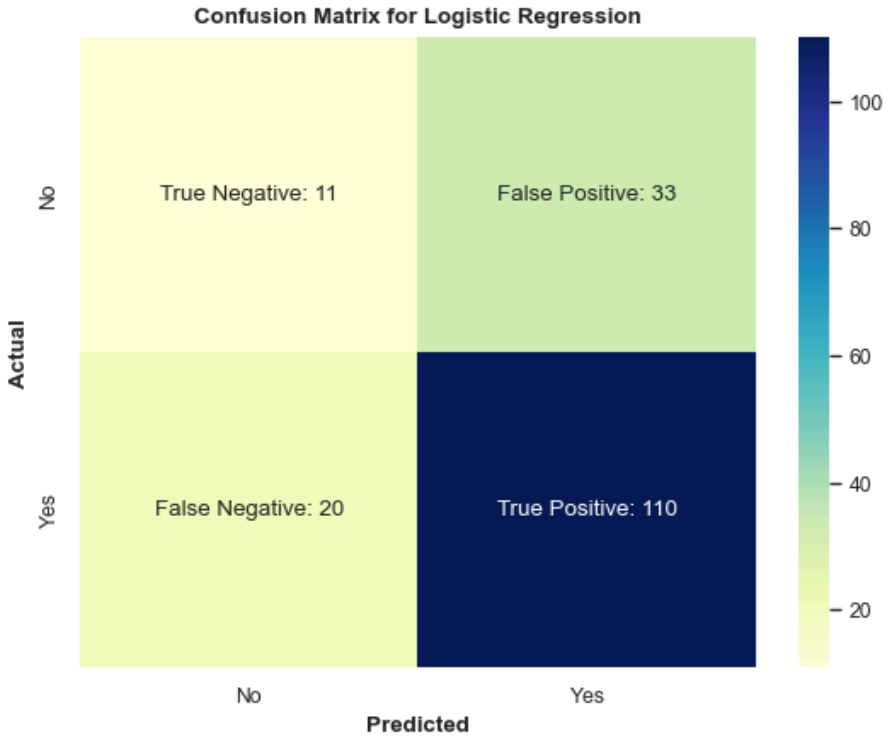
The accuracy of the logistic regression model is 70.11%. The confusion matrix is as follows:

Confusion matrix

In [ ]:
```
confusion_matrix_lr = confusion_matrix(y_test, y_pred_lr)
labels = numpy.reshape([f"True Negative: {confusion_matrix_lr[0, 0]}",
                        f"False Positive: {confusion_matrix_lr[0, 1]}",
                        f"False Negative: {confusion_matrix_lr[1, 0]}",
                        f"True Positive: {confusion_matrix_lr[1, 1]}"], (2, 2))
```

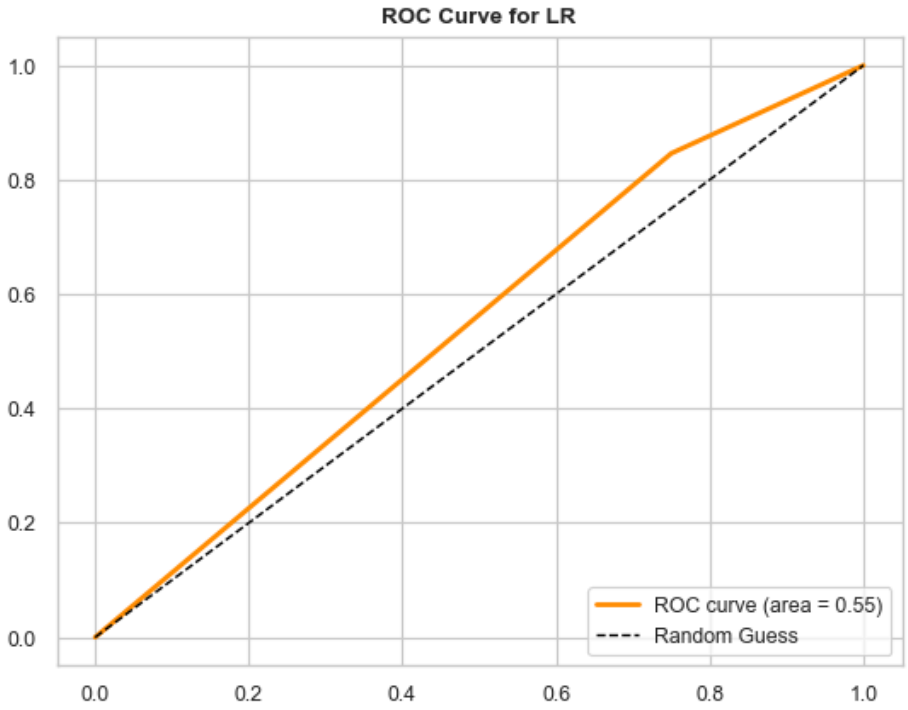
In [ ]:
```
conf_heatmap_lr = seaborn.heatmap(confusion_matrix_lr,
                 annot=labels,
                 cmap="YlGnBu",
                 fmt="",
                 xticklabels=["No", "Yes"],
                 yticklabels=["No", "Yes"])
conf_heatmap_lr.set_xlabel("Predicted", fontdict={'fontweight': 'bold'})
conf_heatmap_lr.set_ylabel("Actual", fontdict={'fontweight': 'bold'})
conf_heatmap_lr.set_title("Confusion Matrix for Logistic Regression", fontdict={'fontweight': 'bo
```

Out[ ]:
```
Text(0.5, 1.0, 'Confusion Matrix for Logistic Regression')
```



ROC Curve and AuC metric

In [ ]:
```
fpr, tpr, thresholds = roc_curve(y_test, y_pred_lr)
roc_auc = roc_auc_score(y_test, y_pred_lr)

plot.figure()
plot.plot(fpr, tpr, color='darkorange', lw=2, label='ROC curve (area = %0.2f)' % roc_auc)
plot.plot([0, 1], [0, 1], color='black', lw=1, linestyle='--', label='Random Guess')
plot.legend(loc='lower right')
plot.grid(True)
plot.title("ROC Curve for LR", fontdict = {"fontweight": "bold"})
plot.show()
```



Support Vector Machine

In [ ]:
```
svm = SVC(kernel="rbf")
svm.fit(X_train, y_train)

y_pred_svm = svm.predict(X_test)
```

Accuracy

In [ ]:
```
accuracy_svm = accuracy_score(y_test, y_pred_svm)
accuracy_svm
```

Out[ ]:
```
0.7126436781609196
```

The accuracy of the logistic regression model is 71.2%. The confusion matrix is as follows:

Confusion matrix

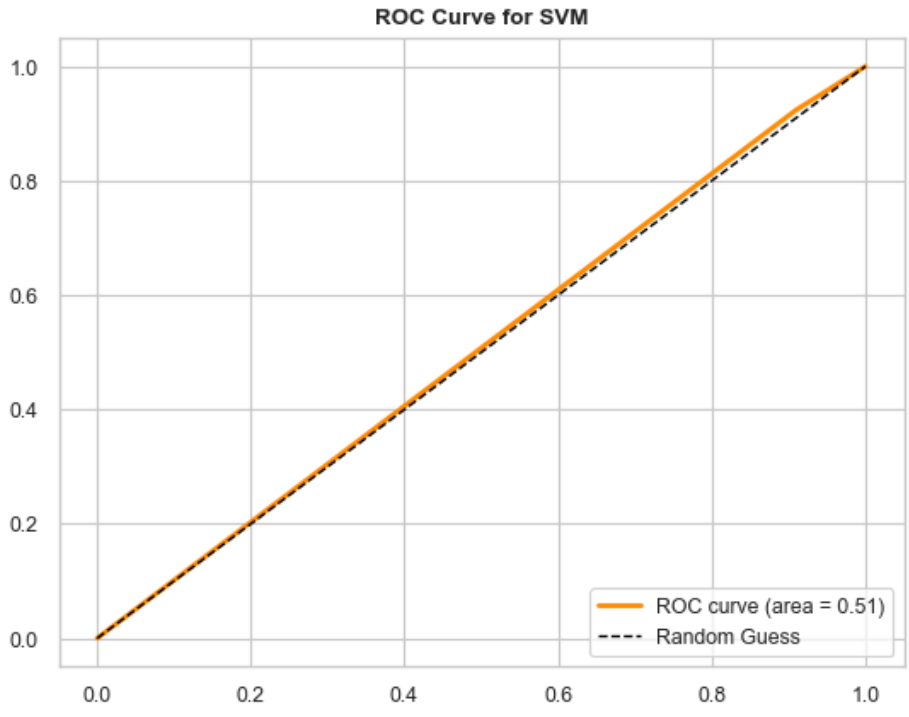
In [ ]:
```
confusion_matrix_svm = confusion_matrix(y_test, y_pred_svm)
labels = numpy.reshape([f"True Negative: {confusion_matrix_svm[0, 0]}",
                        f"False Positive: {confusion_matrix_svm[0, 1]}",
                        f"False Negative: {confusion_matrix_svm[1, 0]}",
                        f"True Positive: {confusion_matrix_svm[1, 1]}"], (2, 2))
```

In [ ]:
```
conf_heatmap_svm = seaborn.heatmap(confusion_matrix_svm,
                annot=labels,
                cmap="rocket",
                fmt="",
                xticklabels=["No", "Yes"],
                yticklabels=["No", "Yes"])
conf_heatmap_svm.set_xlabel("Predicted", fontdict={'fontweight': 'bold'})
conf_heatmap_svm.set_ylabel("Actual", fontdict={'fontweight': 'bold'})
conf_heatmap_svm.set_title("Confusion Matrix for SVM", fontdict={'fontweight': 'bold'})
```

Out[ ]:
```
Text(0.5, 1.0, 'Confusion Matrix for SVM')
```

ROC Curve and AuC metric

In [ ]:
```
fpr, tpr, thresholds = roc_curve(y_test, y_pred_svm)
roc_auc_svm = roc_auc_score(y_test, y_pred_svm)

plot.figure()
plot.plot(fpr, tpr, color='darkorange', lw=2, label='ROC curve (area = %0.2f)' % roc_auc_svm)
plot.plot([0, 1], [0, 1], color='black', lw=1, linestyle='--', label='Random Guess')
plot.legend(loc='lower right')
plot.grid(True)
plot.title("ROC Curve for SVM", fontdict = {"fontweight": "bold"})
plot.show()
```



Conclusion

As we can see from the above modelling, in its current form, the ILP Dataset results in models that are *average* in accuracy. The data appears to be linearly separable, due to which Logistic Regression and SVM (RBF kernel) models result in the same accuracy. **It is noteworthy that the AUC for the SVM model is *slightly worse* at 0.51!**

This indicates that unintentional class-imablance in clinical data collection - such as in this dataset - might often result in biased or inconclusive evidence for treatments and patient outcomes. To combat this, we can employ techniques like SMOTE to oversample the minority class (*Female* participants), which might lead to improvements in model performance.

References

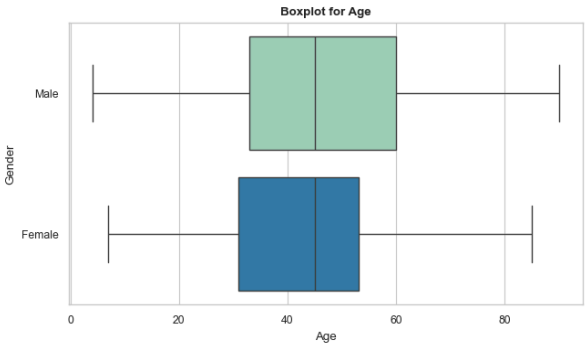
1. Ramana,Bendi and Venkateswarlu,N.. (2012). ILPD (Indian Liver Patient Dataset). UCI Machine Learning Repository. https://doi.org/10.24432/C5D02C.

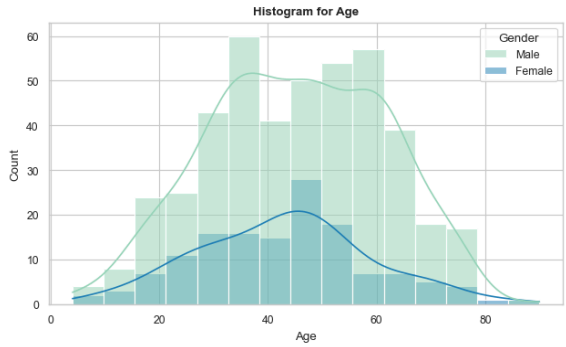
2. Straw, I., & Wu, H. (2022). Investigating for bias in healthcare algorithms: a sex-stratified analysis of supervised machine learning models in liver disease prediction. BMJ Health & Care Informatics, 29.

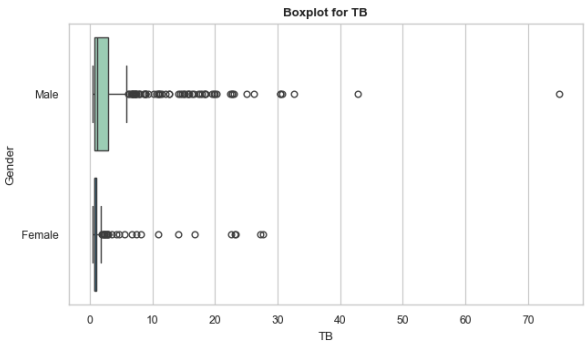
Appendix: individual plots for the presentation

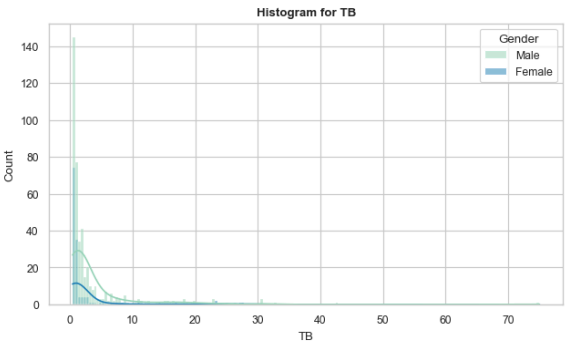
In [ ]:
```
exclude = ["Gender", "Selector"]
continuous = [col for col in liver_data.columns if col not in exclude]

for column in continuous:
    figure, axes = plot.subplots(1, 2, figsize=(16, 4))
    axes = axes.flatten()
    for i, axes in enumerate(axes):
        # Plot boxplot
        if i % 2 == 0:
            seaborn.boxplot(x=liver_data[column], ax=axes, palette = 'YlGnBu', y="Gender", data=l
            axes.set_title(f'Boxplot for {column}', fontdict={'fontweight': 'bold'})
        else:
            seaborn.histplot(x=liver_data[column], ax=axes, kde=True, hue="Gender", palette = 'Yl
            axes.set_title(f'Histogram for {column}', fontdict={'fontweight': 'bold'})

# figure, axes = plot.subplots(len(continuous), 2, figsize=(15, 3 * len(continuous)))
# axes = axes.flatten()

# for i, axes in enumerate(axes):
#     # Plot boxplot
#     if i % 2 == 0:
#         seaborn.boxplot(x=liver_data[continuous[i // 2]], ax=axes, palette = 'YlGnBu', y="Gende
#         axes.set_title(f'Boxplot for {continuous[i // 2]}', fontdict={'fontweight': 'bold'})
#     # # Plot histogram
#     else:
#         seaborn.histplot(x=liver_data[continuous[i // 2]], ax=axes, kde=True, hue="Gender", pal
#         axes.set_title(f'Histogram for {continuous[i // 2]}', fontdict={'fontweight': 'bold'})

# plot.subplots_adjust(hspace=0.5)  # Adjust the spacing between rows
```

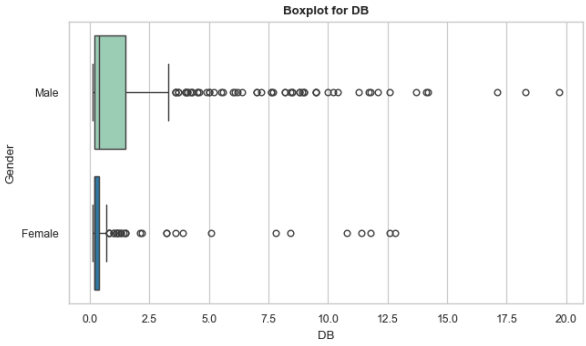
```
C:\Users\utkar\AppData\Local\Temp\ipykernel_20640\3323258243.py:10: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the
`y` variable to `hue` and set `legend=False` for the same effect.

  seaborn.boxplot(x=liver_data[column], ax=axes, palette = 'YlGnBu', y="Gender", data=liver_data)
C:\Users\utkar\AppData\Local\Temp\ipykernel_20640\3323258243.py:10: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the
`y` variable to `hue` and set `legend=False` for the same effect.

  seaborn.boxplot(x=liver_data[column], ax=axes, palette = 'YlGnBu', y="Gender", data=liver_data)
C:\Users\utkar\AppData\Local\Temp\ipykernel_20640\3323258243.py:10: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the
`y` variable to `hue` and set `legend=False` for the same effect.

  seaborn.boxplot(x=liver_data[column], ax=axes, palette = 'YlGnBu', y="Gender", data=liver_data)
C:\Users\utkar\AppData\Local\Temp\ipykernel_20640\3323258243.py:10: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the
`y` variable to `hue` and set `legend=False` for the same effect.

  seaborn.boxplot(x=liver_data[column], ax=axes, palette = 'YlGnBu', y="Gender", data=liver_data)
C:\Users\utkar\AppData\Local\Temp\ipykernel_20640\3323258243.py:10: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the
`y` variable to `hue` and set `legend=False` for the same effect.

  seaborn.boxplot(x=liver_data[column], ax=axes, palette = 'YlGnBu', y="Gender", data=liver_data)
C:\Users\utkar\AppData\Local\Temp\ipykernel_20640\3323258243.py:10: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the
`y` variable to `hue` and set `legend=False` for the same effect.

  seaborn.boxplot(x=liver_data[column], ax=axes, palette = 'YlGnBu', y="Gender", data=liver_data)
C:\Users\utkar\AppData\Local\Temp\ipykernel_20640\3323258243.py:10: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the
`y` variable to `hue` and set `legend=False` for the same effect.

  seaborn.boxplot(x=liver_data[column], ax=axes, palette = 'YlGnBu', y="Gender", data=liver_data)
C:\Users\utkar\AppData\Local\Temp\ipykernel_20640\3323258243.py:10: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the
`y` variable to `hue` and set `legend=False` for the same effect.

  seaborn.boxplot(x=liver_data[column], ax=axes, palette = 'YlGnBu', y="Gender", data=liver_data)
C:\Users\utkar\AppData\Local\Temp\ipykernel_20640\3323258243.py:10: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the
`y` variable to `hue` and set `legend=False` for the same effect.

  seaborn.boxplot(x=liver_data[column], ax=axes, palette = 'YlGnBu', y="Gender", data=liver_data)
C:\Users\utkar\AppData\Local\Temp\ipykernel_20640\3323258243.py:10: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the
`y` variable to `hue` and set `legend=False` for the same effect.

  seaborn.boxplot(x=liver_data[column], ax=axes, palette = 'YlGnBu', y="Gender", data=liver_data)
```

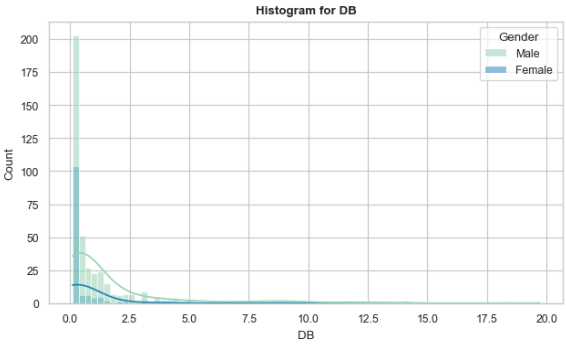

Boxplot for Age
Male
Female
Gender
0
20
40
60
80
Age

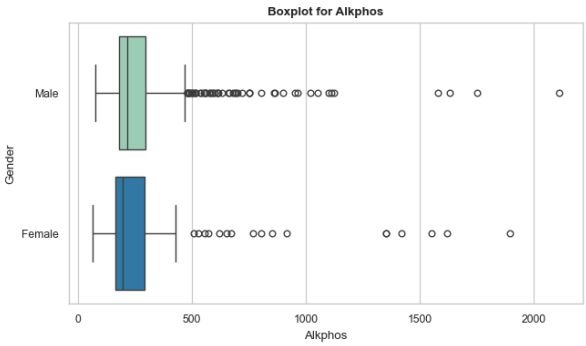

Histogram for Age
Gender
Male
Female
Count
0
10
20
30
40
50
60
Age

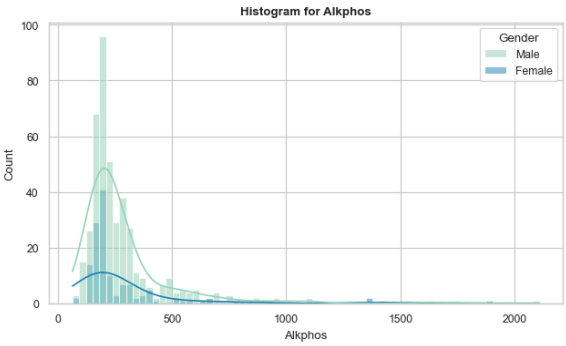

Boxplot for TB
Male
Female
Gender
0
10
20
30
40
50
60
70
TB

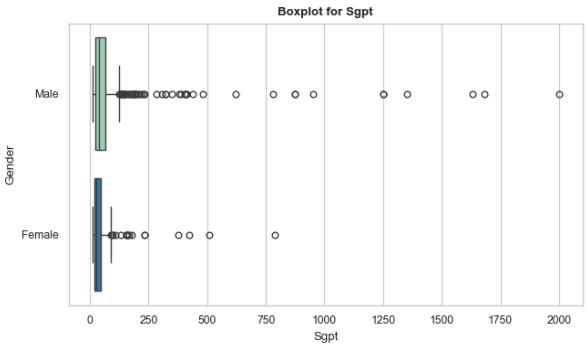

Histogram for TB
Gender
Male
Female
Count
0
20
40
60
80
100
120
140
0
10
20
30
40
50
60
70
TB

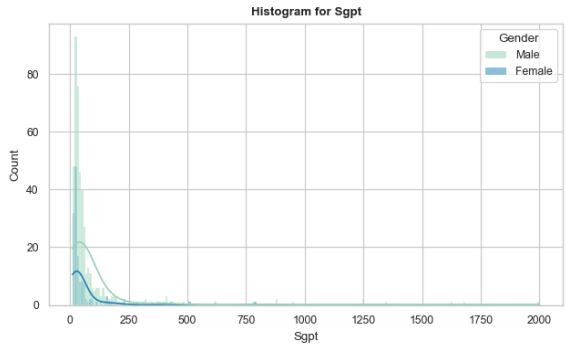

Boxplot for DB
Male
Female
Gender
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
DB

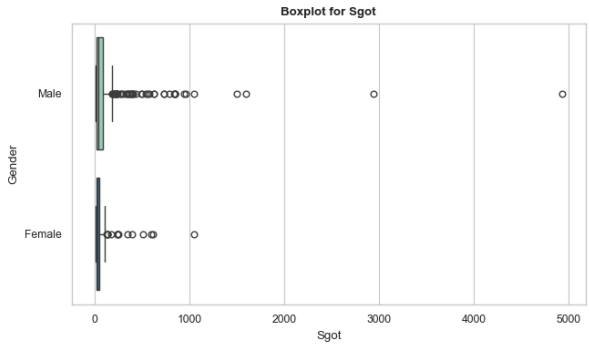

Histogram for DB
Gender
Male
Female
Count
0
25
50
75
100
125
150
175
200
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
DB

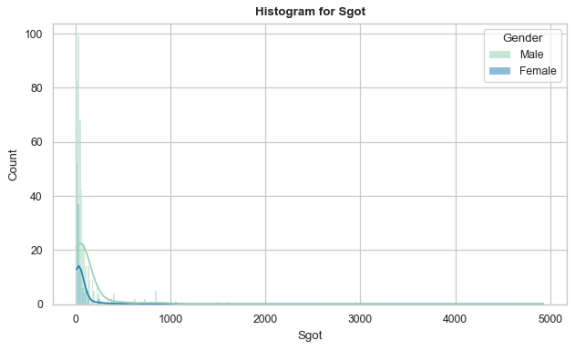

Boxplot for Alkphos
Male
Female
Gender
0
500
1000
1500
2000
Alkphos

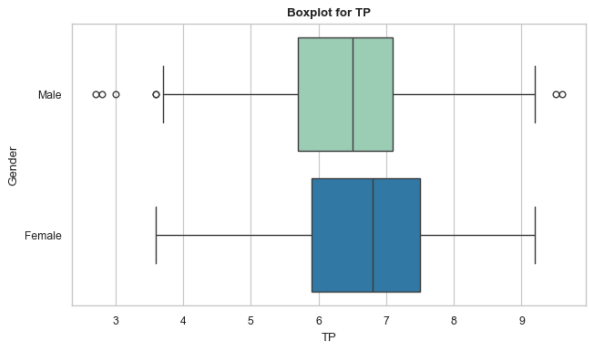

Histogram for Alkphos
Gender
Male
Female
Count
0
20
40
60
80
100
0
500
1000
1500
2000
Alkphos

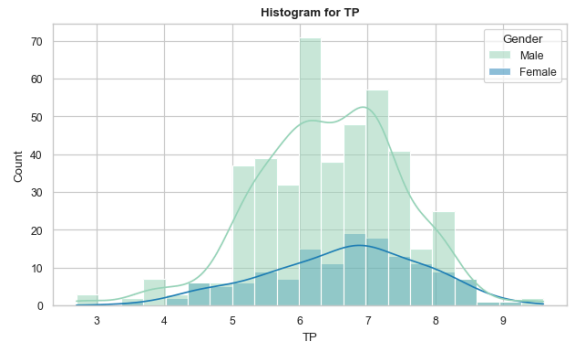

Boxplot for Sgpt
Male
Female
Gender
0
250
500
750
1000
1250
1500
1750
2000
Sgpt

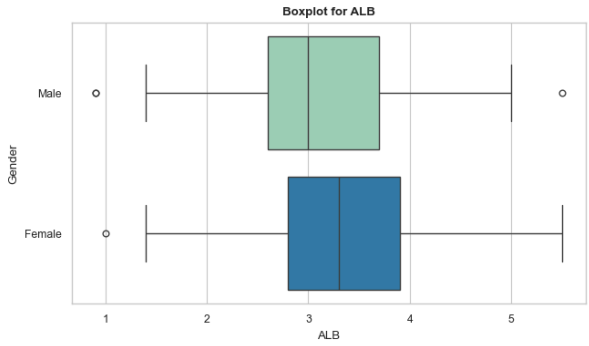

Histogram for Sgpt
Gender
Male
Female
Count
0
20
40
60
80
0
250
500
750
1000
1250
1500
1750
2000
Sgpt

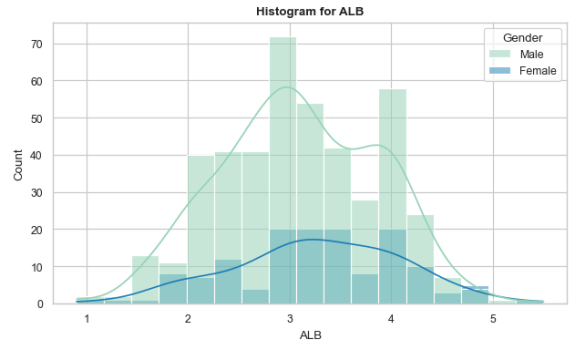

Boxplot for Sgot
Male
Female
Gender
0
1000
2000
3000
4000
5000
Sgot

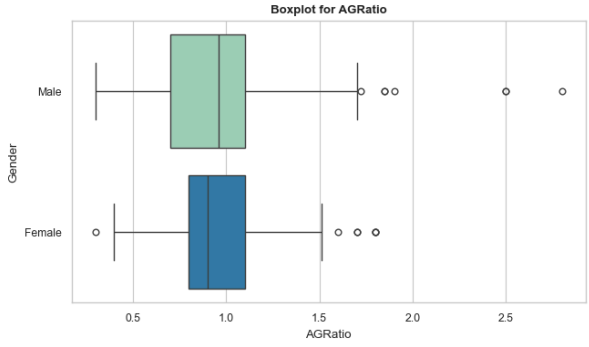

Histogram for Sgot
Gender
Male
Female
Count
0
20
40
60
80
100
0
1000
2000
3000
4000
5000
Sgot

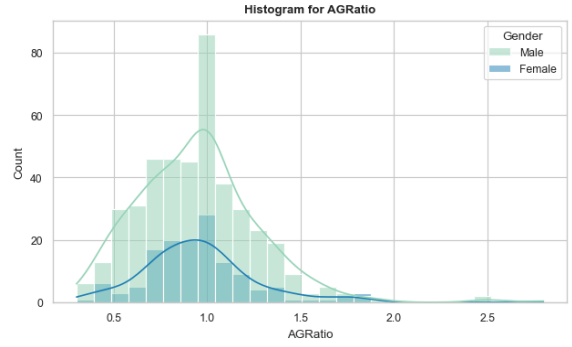

Boxplot for TP
Male
Female
Gender
3
4
5
6
7
8
9
TP

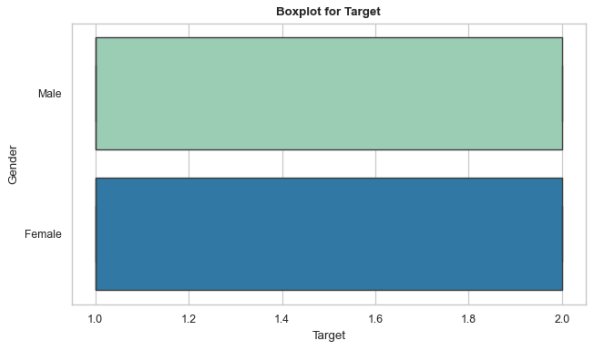

Histogram for TP
Gender
Male
Female
Count
0
10
20
30
40
50
60
70
3
4
5
6
7
8
9
TP

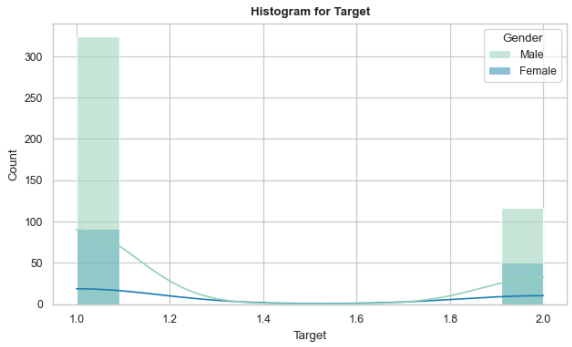

Boxplot for ALB
Male
Female
Gender
1
2
3
4
5
ALB


Histogram for ALB
Gender
Male
Female
Count
0
10
20
30
40
50
60
70
1
2
3
4
5
ALB


Boxplot for AGRatio
Male
Female
Gender
0.5
1.0
1.5
2.0
2.5
AGRatio


Histogram for AGRatio
Gender
Male
Female
Count
0
20
40
60
80
0.5
1.0
1.5
2.0
2.5
AGRatio


Boxplot for Target
Male
Female
Gender
1.0
1.2
1.4
1.6
1.8
2.0
Target


Histogram for Target
Gender
Male
Female
Count
0
50
100
150
200
250
300
1.0
1.2
1.4
1.6
1.8
2.0
Target