Project Title : Classification Data Set (Hepatitis C Prediction)

(Group 2)

- Sindhu Vempada
- Bakht Singh Basaram - 11546730
- Carmen suet yee wong - 11510519
- Anh Hai Ha - 11541845
- Shanmukha Nadh

Data Set Information:

The HCV dataset was obtained from the University of California at Irvine (UCI) Machine Learning Repository https://archive.ics.uci.edu/ml/datasets/HCV+data (https://archive.ics.uci.edu/ml/datasets/HCV+data). This dataset comprised clinical laboratory and demographic data of 615 blood donors and patients with hepatitis C. A total of 13 features were defined for each patient record. These features are tests like:

- albumin (ALB)
- alkaline phosphatase (ALP)
- bilirubin (BIL)
- cholesterol (CHOL)
- creatinine blood test (CREA)
- choline esterase (CHE)
- γ-glutamyl-transferase (GGT)
- aspartate aminotransferase (AST)
- alanine aminotransferase (ALT)
- total protein test (PROT)

The demographic characteristics included age and sex. In this dataset, the final diagnosis was characterized by five outcomes of interest: blood donors, suspected blood donors, hepatitis C, fibrosis, and cirrhosis. The patients diagnosed with HCV ranged from chronic hepatitis C infection without fibrosis to end-stage liver cirrhosis with a need for LTx.

Note:

1. Most of the comments mentioned and procedure followed in this notebook are from a research article https://www.sciencedirect.com/science/article/pii/S266710262200002X#bib0028 (https://www.sciencedirect.com/science/article/pii/S266710262200002X#bib0028)
2. Some part of the code is reused from https://www.kaggle.com/code/calebreigada/liver-disease-analysis-eda-smote-optuna-shap/notebook (https://www.kaggle.com/code/calebreigada/liver-disease-analysis-eda-smote-optuna-shap/notebook)

In [1]:
```
#Import needed libraries
import numpy as np #linear algebra
import pandas as pd #data manipulation
import matplotlib.pyplot as plt #data viz
from matplotlib.gridspec import GridSpec #data viz
from matplotlib.animation import FuncAnimation #animation
import seaborn as sns #data viz

#data preprocessing
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.impute import KNNImputer
from sklearn.impute import SimpleImputer
from sklearn.pipeline import Pipeline

from sklearn.decomposition import PCA #principle component analysis
from IPython.display import HTML #display gif
from sklearn.model_selection import train_test_split
from imblearn.over_sampling import SMOTE #balance classes
import optuna

#models to try
from sklearn.ensemble import RandomForestClassifier, VotingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB

#graph of model
import graphviz

#metrics
from sklearn import metrics

#shap
import shap
import warnings #warnings
warnings.filterwarnings('ignore') #Hides warning popups
```

In [2]:
```
#loading the dataset
data = pd.read_csv('train_classification.csv')
```

To quickly understand the summary of Data

In [3]:
```
data.head() # top 5 rows
```

Out[3]:

	Age	Sex	ALB	ALP	ALT	AST	BIL	CHE	CHOL	CREA	GGT	PROT	Category
0	39	f	46.4	59.2	14.1	18.9	4.5	7.90	4.55	61.0	14.5	77.3	0=Blood Donor
1	37	m	46.1	44.3	42.7	26.5	6.4	10.86	5.05	74.0	22.2	73.1	0=Blood Donor
2	32	m	50.9	65.5	23.2	21.2	6.9	8.69	4.10	83.0	13.7	71.3	0=Blood Donor
3	46	f	36.7	62.3	10.8	17.4	3.7	6.17	4.07	67.0	15.1	69.0	0=Blood Donor
4	56	m	23.0	105.6	5.1	123.0	43.0	1.80	2.40	62.7	35.9	62.8	3=Cirrhosis

In [4]:
```
data.tail() # last 5 rows
```

Out[4]:

	Age	Sex	ALB	ALP	ALT	AST	BIL	CHE	CHOL	CREA	GGT	PROT	Category
548	60	m	40.4	46.8	17.7	25.7	13.5	5.79	5.42	92.0	19.2	70.0	0=Blood Donor
549	32	m	42.4	86.3	20.3	20.0	35.2	5.46	4.45	81.0	15.9	69.9	0=Blood Donor
550	38	f	40.0	73.5	16.6	19.2	8.3	5.23	5.52	54.0	24.0	71.0	0=Blood Donor
551	49	m	44.3	84.1	29.0	29.0	16.2	8.18	4.65	87.0	21.9	70.8	0=Blood Donor
552	58	m	43.0	99.1	12.2	63.2	13.0	5.95	6.15	147.3	491.0	65.6	1=Hepatitis

In [5]:
```
data.shape # to know number of columns,rows
```

Out[5]:
```
(553, 13)
```

In [6]:
```
#check details about the data set
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 553 entries, 0 to 552
Data columns (total 13 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   Age       553 non-null    int64  
 1   Sex       553 non-null    object 
 2   ALB       552 non-null    float64
 3   ALP       536 non-null    float64
 4   ALT       552 non-null    float64
 5   AST       553 non-null    float64
 6   BIL       553 non-null    float64
 7   CHE       553 non-null    float64
 8   CHOL      544 non-null    float64
 9   CREA      553 non-null    float64
 10  GGT       553 non-null    float64
 11  PROT      552 non-null    float64
 12  Category  553 non-null    object 
dtypes: float64(10), int64(1), object(2)
memory usage: 56.3+ KB
```

In [7]:
```
#To know the statistical description of the dataset
data.describe().T
```

Out[7]:

	count	mean	std	min	25%	50%	75%	max
Age	553.0	47.459313	10.202420	19.00	39.0000	47.00	55.000	77.00
ALB	552.0	41.641123	5.843118	14.90	38.8750	42.00	45.225	82.20
ALP	536.0	67.962127	26.643695	11.30	52.2750	65.55	80.125	416.60
ALT	552.0	28.574457	25.308527	0.90	16.4750	23.00	33.200	325.30
AST	553.0	34.576492	30.292877	10.60	21.7000	26.20	33.000	319.80
BIL	553.0	11.314828	18.839697	1.80	5.4000	7.30	11.500	254.00
CHE	553.0	8.170253	2.196384	1.42	6.9400	8.18	9.570	16.41
CHOL	544.0	5.360809	1.123139	1.43	4.6175	5.30	6.060	9.67
CREA	553.0	79.782098	30.358409	8.00	67.0000	77.00	88.000	519.00
GGT	553.0	38.938336	53.895354	4.50	15.8000	23.30	40.100	650.90
PROT	552.0	71.962500	5.506352	44.80	69.2000	72.00	75.325	90.00

In [8]:
```
# to change the position[last to first] of category column
col = data.pop("Category")
data.insert(0,col.name,col)
```

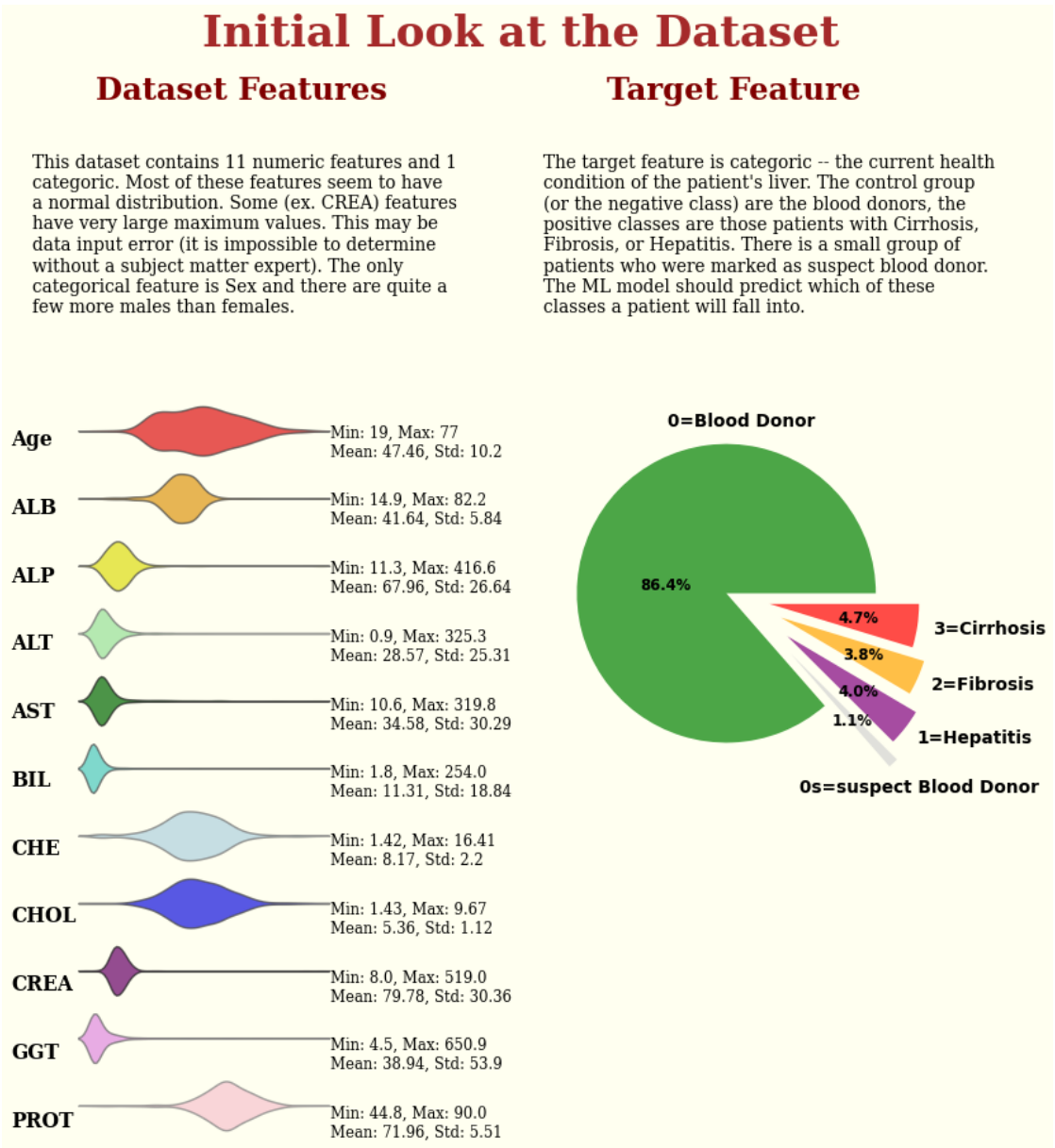
In [9]:
```
data.head()
```

Out[9]:

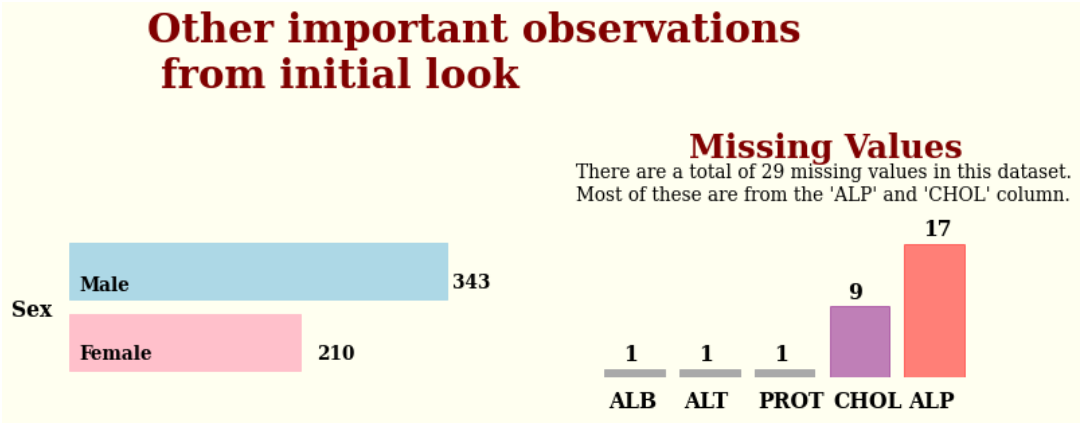
	Category	Age	Sex	ALB	ALP	ALT	AST	BIL	CHE	CHOL	CREA	GGT	PROT
0	0=Blood Donor	39	f	46.4	59.2	14.1	18.9	4.5	7.90	4.55	61.0	14.5	77.3
1	0=Blood Donor	37	m	46.1	44.3	42.7	26.5	6.4	10.86	5.05	74.0	22.2	73.1
2	0=Blood Donor	32	m	50.9	65.5	23.2	21.2	6.9	8.69	4.10	83.0	13.7	71.3
3	0=Blood Donor	46	f	36.7	62.3	10.8	17.4	3.7	6.17	4.07	67.0	15.1	69.0
4	3=Cirrhosis	56	m	23.0	105.6	5.1	123.0	43.0	1.80	2.40	62.7	35.9	62.8

In [10]:
```
# Basic Statistics Visualization of Dataset

fig = plt.figure(figsize = (24,12), dpi = 70)
gs = GridSpec(ncols=10, nrows=12, left=0.05, right=0.5, wspace=0.2, hspace=0.1)
fig.patch.set_facecolor('ivory')

ax1_1 = fig.add_subplot(gs[1:2, 1:4]) #top left
ax1_2 = fig.add_subplot(gs[2:3, 1:4])
ax1_3 = fig.add_subplot(gs[3:4, 1:4])
ax1_4 = fig.add_subplot(gs[4:5, 1:4])
ax1_5 = fig.add_subplot(gs[5:6, 1:4])
ax1_6 = fig.add_subplot(gs[6:7, 1:4])
ax1_7 = fig.add_subplot(gs[7:8, 1:4])
ax1_8 = fig.add_subplot(gs[8:9, 1:4])
ax1_9 = fig.add_subplot(gs[9:10, 1:4])
ax1_10 = fig.add_subplot(gs[10:11, 1:4])
ax1_11 = fig.add_subplot(gs[11:12, 1:4])


ax2 = fig.add_subplot(gs[:8, 6:]) #top right(8rows,5cols)

# axes list
axes = [ ax1_1, ax1_2, ax1_3, ax1_4, ax1_5, ax1_6,
        ax1_7, ax1_8, ax1_9, ax1_10, ax1_11, ax2]


# setting of axes; visibility of axes and spines turn off
for ax in axes:
    ax.axes.get_yaxis().set_visible(False)
    ax.axes.get_xaxis().set_visible(False)
    ax.set_facecolor('ivory')

    for loc in ['left', 'right', 'top', 'bottom']:
        ax.spines[loc].set_visible(False)

#-------------------------------------------------------------
#Plots violin charts of numeric columns in top left
top_left_axes = [ax1_1, ax1_2, ax1_3, ax1_4, ax1_5, ax1_6,
        ax1_7, ax1_8, ax1_9, ax1_10, ax1_11]
colors = ['red', 'orange', 'yellow', 'lightgreen', 'green', 'turquoise', 'lightblue',
          'blue', 'purple', 'violet', 'pink']
y = 0.775 #0.805
for ax, color, column in zip(top_left_axes, colors, list(data.describe().columns)):
    sns.violinplot(x=column, y=None, data=data, ax=ax, inner=None, color=color)
    ax.collections[::2][0].set_alpha(0.75)
    fig.text(0.07, y, column,
             {'font':'Serif', 'weight':'bold','color': 'black', 'size':16})
    stats_text = "Min: {}, Max: {}\nMean: {}, Std: {}".format(round(data[column].min(), 2),
                                                              round(data[column].max(), 2),
                                                              round(data[column].mean(), 2),
                                                              round(data[column].std(), 2))
    fig.text(0.22, y-0.01, stats_text,
             {'font':'Serif', 'weight':'normal','color': 'black', 'size':12})
    y-=0.0640 #0.0475
#ax1 title
fig.text(0.11, 1.1, 'Dataset Features', {'font':'Serif', 'weight':'bold','color': 'maroon', 'size':25})
fig.text(0.08, 0.9, "This dataset contains 11 numeric features and 1\ncategoric." +\
         " Most of these features seem to have\na normal distribution. " +\
         "Some (ex. CREA) features\nhave very large maximum values. " +\
         "This may be\ndata input error (it is impossible to determine\nwithout a subject matter expert). " +\
         "The only\ncategorical feature is Sex and there are quite a\nfew more males than females.",
        {'font':'Serif', 'weight':'normal','color': 'black', 'size':14})

#-------------------------------------------------------------
#plots split of target feature in pie chart
ax2_plot = ax2.pie(data.groupby('Category').Category.count().values,
                   labels=data.groupby('Category').Category.count().index,
                    autopct='%1.1f%%', explode=[0.1, 0.5, 0.4, 0.3, 0.2],
                   colors=['green', 'lightgrey', 'purple', 'orange', 'red'])

for piece in ax2_plot[0]:
    piece.set_alpha(0.7)

for i, text in enumerate(ax2_plot[1]):
    text.set_weight('bold')
    text.set_size(14)
    if i == 0:
        text.set_y(1.2)
        text.set_x(0.5)
    if i == 1:
        text.set_y(-1.25)
        text.set_x(0.4)
for i, text in enumerate(ax2_plot[2]):
    text.set_weight('bold')
    text.set_size(12)
```

```
    if i == 0:
        text.set_y(0.1)
        text.set_x(-0.5)

fig.text(0.35, 1.1, 'Target Feature',  {'font':'Serif', 'weight':'bold','color': 'maroon', 'size':25})
fig.text(0.32, 0.9,
        "The target feature is categoric -- the current health\ncondition of the patient's liver. " +\
         "The control group\n(or the negative class) are the blood donors, the\npositive classes " +\
         "are those patients with Cirrhosis,\nFibrosis, or Hepatitis. There is a small group of " +\
         "\npatients who were marked as suspect blood donor.\nThe ML model should predict which of these\n" +\
         "classes a patient will fall into.",
        {'font':'Serif', 'weight':'normal','color': 'black', 'size':14})

#-----------------------------------------------------------------
fig.text(0.16,1.15,'Initial Look at the Dataset', {'font':'Serif', 'weight':'bold','color': 'brown', 'size':35})
plt.show()
```



In [11]:
```
# Visualizes Basic Data Statistics

fig = plt.figure(figsize = (24,3), dpi = 70)
gs = GridSpec(ncols=10, nrows=3, left=0.05, right=0.5, wspace=0.2, hspace=0.1)
fig.patch.set_facecolor('ivory')

ax1 = fig.add_subplot(gs[1:3, 1:5])
ax2 = fig.add_subplot(gs[1:3, 6:])

# axes list
axes = [ ax1,ax2]

# setting of axes; visibility of axes and spines turn off
for ax in axes:
    ax.axes.get_yaxis().set_visible(False)
    ax.axes.get_xaxis().set_visible(False)
    ax.set_facecolor('ivory')

    for loc in ['left', 'right', 'top', 'bottom']:
        ax.spines[loc].set_visible(False)
#-----------------------------------------------------------
#plots sex in bottom left
sex_data = data.groupby('Sex').Sex.count()
ax1_plot = ax1.barh(sex_data.index, sex_data.values)
for i, rect in enumerate(ax1_plot):
    if i == 0:
        rect.set_color('pink')
    else:
        rect.set_color('lightblue')


fig.text(0.07, 0.34, 'Sex', {'font':'Serif', 'weight':'bold','color': 'black', 'size':16})
fig.text(0.1, 0.43, 'Male', {'font':'Serif', 'weight':'bold','color': 'black', 'size':14})
fig.text(0.1, 0.1875, 'Female', {'font':'Serif', 'weight':'bold','color': 'black', 'size':14})
fig.text(0.265, 0.44, sex_data['m'], {'font':'Serif', 'weight':'bold','color': 'black', 'size':14})
fig.text(0.205, 0.1875, sex_data['f'], {'font':'Serif', 'weight':'bold','color': 'black', 'size':14})

#-----------------------------------------------------------
#plot null values in barchart
null_data = data.isnull().sum()[data.isnull().sum() > 0].sort_values()
ax2_plot = ax2.bar(null_data.index, null_data.values)
for i, rect in enumerate(ax2_plot):
    height = rect.get_height()
    ax2.text((rect.get_x() + (rect.get_width() / 2.)) - 0.35, -4, null_data.index[i],
           {'font':'Serif', 'weight':'bold','color': 'black', 'size':16})
    ax2.text((rect.get_x() + (rect.get_width() / 2.)) - 0.15, 1. + height, null_data.values[i],
           {'font':'Serif', 'weight':'bold','color': 'black', 'size':16})
    if i < 3:
        rect.set_color('darkgrey')
    elif i == 3:
        rect.set_color('purple')
        rect.set_alpha(0.5)
    else:
        rect.set_color('red')
        rect.set_alpha(0.5)
fig.text(0.37, 0.9, 'Missing Values', {'font':'Serif', 'weight':'bold','color': 'maroon', 'size':25})
fig.text(0.32, 0.75,
        "There are a total of 29 missing values in this dataset.\n" +\
         "Most of these are from the 'ALP' and 'CHOL' column.",
        {'font':'Serif', 'weight':'normal','color': 'black', 'size':14})

#-----------------------------------------------------------------
fig.text(0.13,1.15,'Other important observations\n from initial look', {'font':'Serif', 'weight':'bold','color': 'maro
plt.show()
```



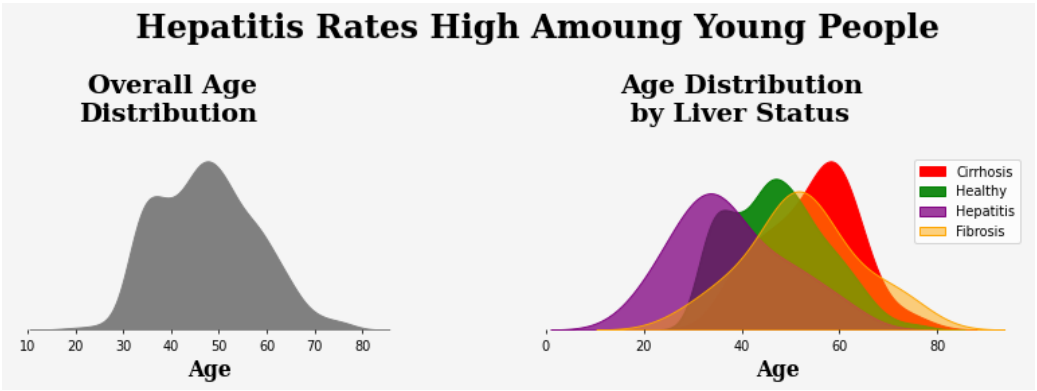
In [12]:
```
# maps target feature
# suspects are binned with blood donors for simplicity

target_map = {'0=Blood Donor': 'Healthy',
              '0s=suspect Blood Donor': 'Healthy',
              '1=Hepatitis': 'Hepatitis',
              '2=Fibrosis': 'Fibrosis',
              '3=Cirrhosis': 'Cirrhosis'}

data['Category'] = data.Category.map(target_map)
```

Univariate Analysis of all features

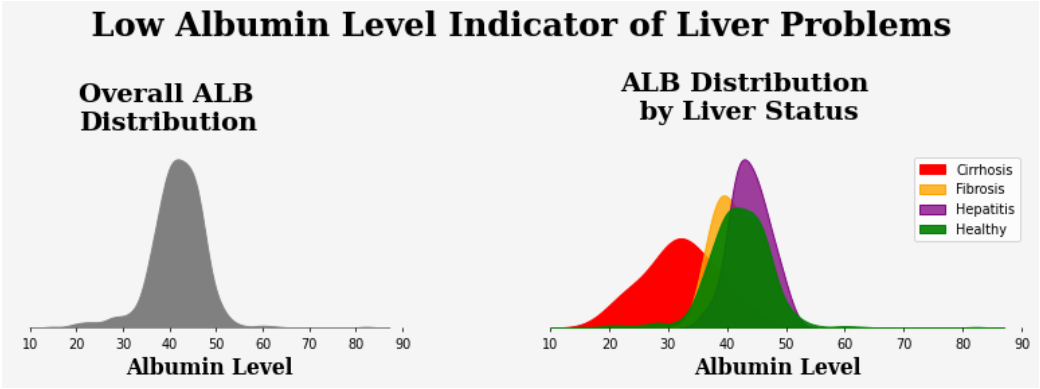
Feature : 'Age'

Description : Age of patient (in Years)

In [13]:
```
# plots univariate analysis of age

fig = plt.figure(figsize = (24,4), dpi = 70)
gs = GridSpec(ncols=10, nrows=8, left=0.05, right=0.5, wspace=0.2, hspace=0.1)
fig.patch.set_facecolor('#f5f5f5')
sns.set_palette(sns.color_palette(['green','purple', 'orange', 'red']))

ax1 = fig.add_subplot(gs[3:8, 0:4])
ax2 = fig.add_subplot(gs[3:8, 5:])

# axes list
axes = [ ax1,ax2]

# setting of axes; visibility of axes and spines turn off
for ax in axes:

    ax.axes.get_yaxis().set_visible(False)
    #ax.axes.get_xaxis().set_visible(False)
    ax.set_facecolor('#f5f5f5')

    for loc in ['left', 'right', 'top', 'bottom']:
        ax.spines[loc].set_visible(False)



#-------Ax 1--------------------------------------------------
sns.kdeplot('Age', data=data, ax=ax1, shade=True, color='grey', alpha=1)
ax1.set_xlabel('Age', {'font':'Serif', 'fontsize':16,'fontweight':'bold', 'color':'black'})

#-------Ax 2--------------------------------------------------
sns.kdeplot('Age', data=data[data.Category=='Cirrhosis'], ax=ax2, shade=True, color='red', alpha=1, label='Cirrhosis')
sns.kdeplot('Age', data=data[data.Category=='Healthy'], ax=ax2, shade=True, color='green', alpha=0.9, label='Healthy')
sns.kdeplot('Age', data=data[data.Category=='Hepatitis'], ax=ax2, shade=True, color='purple', alpha=0.75, label='Hepati
sns.kdeplot('Age', data=data[data.Category=='Fibrosis'], ax=ax2, shade=True, color='orange', alpha=0.5, label='Fibrosis

ax2.legend()
ax2.set_xlabel('Age', {'font':'Serif', 'fontsize':16,'fontweight':'bold', 'color':'black'})

#-------------------------------------------------------------------------
#axes titles
#ax1
fig.text(0.08, 0.68, " Overall Age\nDistribution",
        {'font':'Serif', 'fontsize':20,'fontweight':'bold', 'color':'black'})
#ax2
fig.text(0.32, 0.68, "Age Distribution\n by Liver Status",
        {'font':'Serif', 'fontsize':20,'fontweight':'bold', 'color':'black'})
#title
fig.text(0.10, 0.9, " Hepatitis Rates High Amoung Young People",
        {'font':'Serif', 'fontsize':25,'fontweight':'bold', 'color':'black'})

plt.show()
```



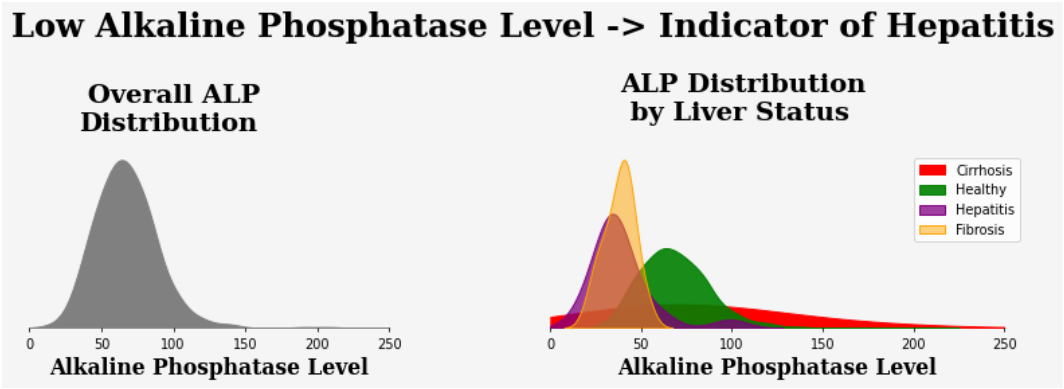
In [14]:
```
# plots univariate analysis of ALB

fig = plt.figure(figsize = (24,4), dpi = 70)
gs = GridSpec(ncols=10, nrows=8, left=0.05, right=0.5, wspace=0.2, hspace=0.1)
fig.patch.set_facecolor('#f5f5f5')
sns.set_palette(sns.color_palette(['green','purple', 'orange', 'red']))

ax1 = fig.add_subplot(gs[3:8, 0:4])
ax2 = fig.add_subplot(gs[3:8, 5:])

# axes list
axes = [ ax1,ax2]

# setting of axes; visibility of axes and spines turn off
for ax in axes:

    ax.axes.get_yaxis().set_visible(False)
    #ax.axes.get_xaxis().set_visible(False)
    ax.set_facecolor('#f5f5f5')

    for loc in ['left', 'right', 'top', 'bottom']:
        ax.spines[loc].set_visible(False)

#-------Ax 1-----------------------------------------------
sns.kdeplot('ALB', data=data, ax=ax1, shade=True, color='grey', alpha=1)
ax1.set_xlabel('Albumin Level', {'font':'Serif', 'fontsize':16,'fontweight':'bold', 'color':'black'})

#-------Ax 2-----------------------------------------------
sns.kdeplot('ALB', data=data[data.Category=='Cirrhosis'], ax=ax2, shade=True, color='red', alpha=1, label='Cirrhosis')
sns.kdeplot('ALB', data=data[data.Category=='Fibrosis'], ax=ax2, shade=True, color='orange', alpha=0.8, label='Fibrosi
sns.kdeplot('ALB', data=data[data.Category=='Hepatitis'], ax=ax2, shade=True, color='purple', alpha=0.75, label='Hepat
sns.kdeplot('ALB', data=data[data.Category=='Healthy'], ax=ax2, shade=True, color='green', alpha=0.9, label='Healthy')


ax2.legend()
ax2.set_xlabel('Albumin Level', {'font':'Serif', 'fontsize':16,'fontweight':'bold', 'color':'black'})

#-------------------------------------------------------------------
#axes titles
#ax1
fig.text(0.08, 0.65, "Overall ALB\nDistribution",
        {'font':'Serif', 'fontsize':20,'fontweight':'bold', 'color':'black'})
#ax2
fig.text(0.32, 0.68, "ALB Distribution\n  by Liver Status",
        {'font':'Serif', 'fontsize':20,'fontweight':'bold', 'color':'black'})
#main figure title
fig.text(0.05, 0.9, "       Low Albumin Level Indicator of Liver Problems",
        {'font':'Serif', 'fontsize':25,'fontweight':'bold', 'color':'black'})

plt.show()
```



Feature : 'ALB'

Description : It measures the amount of albumin in your blood.

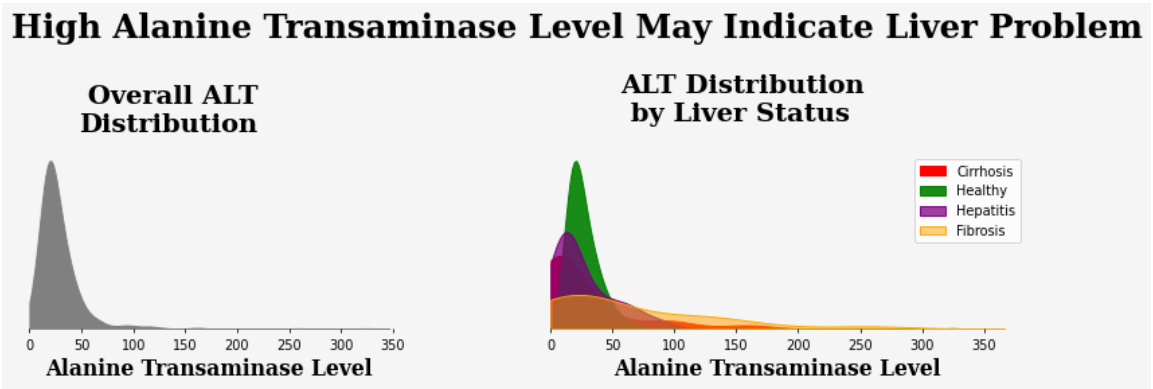
In [15]:
```
# plots univariate analysis of ALP

fig = plt.figure(figsize = (24,4), dpi = 70)
gs = GridSpec(ncols=10, nrows=8, left=0.05, right=0.5, wspace=0.2, hspace=0.1)
fig.patch.set_facecolor('#f5f5f5')
sns.set_palette(sns.color_palette(['green','purple', 'orange', 'red']))

ax1 = fig.add_subplot(gs[3:8, 0:4])
ax2 = fig.add_subplot(gs[3:8, 5:])

# axes list
axes = [ ax1,ax2]

# setting of axes; visibility of axes and spines turn off
for ax in axes:

    ax.axes.get_yaxis().set_visible(False)
    #ax.axes.get_xaxis().set_visible(False)
    ax.set_facecolor('#f5f5f5')

    for loc in ['left', 'right', 'top', 'bottom']:
        ax.spines[loc].set_visible(False)


#-------Ax 1--------------------------------------------------
sns.kdeplot('ALP', data=data, ax=ax1, shade=True, color='grey', alpha=1, clip=[0, 250])
ax1.set_xlabel('Alkaline Phosphatase Level', {'font':'Serif', 'fontsize':16,'fontweight':'bold', 'color':'black'})

#-------Ax 2--------------------------------------------------
sns.kdeplot('ALP', data=data[data.Category=='Cirrhosis'], ax=ax2, shade=True, color='red',
            alpha=1, label='Cirrhosis', clip=[0, 250])
sns.kdeplot('ALP', data=data[data.Category=='Healthy'], ax=ax2, shade=True, color='green',
            alpha=0.9, label='Healthy', clip=[0, 250])
sns.kdeplot('ALP', data=data[data.Category=='Hepatitis'], ax=ax2, shade=True, color='purple',
            alpha=0.75, label='Hepatitis', clip=[0, 250])
sns.kdeplot('ALP', data=data[data.Category=='Fibrosis'], ax=ax2, shade=True, color='orange',
            alpha=0.5, label='Fibrosis', clip=[0, 250])

ax2.legend()
ax2.set_xlabel('Alkaline Phosphatase Level', {'font':'Serif', 'fontsize':16,'fontweight':'bold', 'color':'black'})

#-------------------------------------------------------
#axes titles
#ax1
fig.text(0.08, 0.65, " Overall ALP\nDistribution",
        {'font':'Serif', 'fontsize':20,'fontweight':'bold', 'color':'black'})
#ax2
fig.text(0.32, 0.68, "ALP Distribution\n by Liver Status",
        {'font':'Serif', 'fontsize':20,'fontweight':'bold', 'color':'black'})
#figure title
fig.text(0.05, 0.9, "Low Alkaline Phosphatase Level -> Indicator of Hepatitis",
        {'font':'Serif', 'fontsize':25,'fontweight':'bold', 'color':'black'})

plt.show()
```



Feature : 'ALP'

Description : It measures the amount of alkaline phosphatase enzyme in your bloodstream..

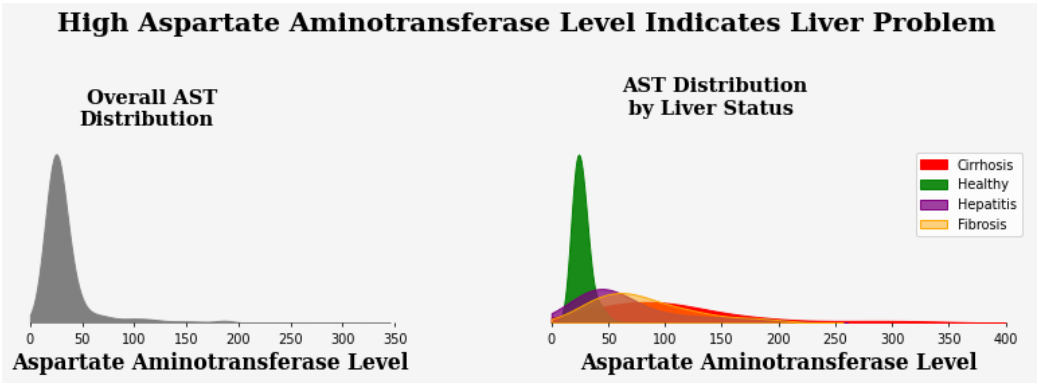
In [16]:
```
# plots univariate analysis of ALT

fig = plt.figure(figsize = (24,4), dpi = 70)
gs = GridSpec(ncols=10, nrows=8, left=0.05, right=0.5, wspace=0.2, hspace=0.1)
fig.patch.set_facecolor('#f5f5f5')
sns.set_palette(sns.color_palette(['green','purple', 'orange', 'red']))

ax1 = fig.add_subplot(gs[3:8, 0:4])
ax2 = fig.add_subplot(gs[3:8, 5:])

# axes list
axes = [ ax1,ax2]

# setting of axes; visibility of axes and spines turn off
for ax in axes:

    ax.axes.get_yaxis().set_visible(False)
    #ax.axes.get_xaxis().set_visible(False)
    ax.set_facecolor('#f5f5f5')

    for loc in ['left', 'right', 'top', 'bottom']:
        ax.spines[loc].set_visible(False)

#-------Ax 1-----------------------------------------------
sns.kdeplot('ALT', data=data, ax=ax1, shade=True, color='grey', alpha=1, clip=[0,500])
ax1.set_xlabel('Alanine Transaminase Level', {'font':'Serif', 'fontsize':16,'fontweight':'bold', 'color':'black'})

#-------Ax 2-----------------------------------------------
sns.kdeplot('ALT', data=data[data.Category=='Cirrhosis'], ax=ax2, shade=True, color='red',
            alpha=1, label='Cirrhosis', clip=[0,500])
sns.kdeplot('ALT', data=data[data.Category=='Healthy'], ax=ax2, shade=True, color='green',
            alpha=0.9, label='Healthy', clip=[0,500])
sns.kdeplot('ALT', data=data[data.Category=='Hepatitis'], ax=ax2, shade=True, color='purple',
            alpha=0.75, label='Hepatitis', clip=[0,500])
sns.kdeplot('ALT', data=data[data.Category=='Fibrosis'], ax=ax2, shade=True, color='orange',
            alpha=0.5, label='Fibrosis', clip=[0,500])

ax2.legend()
ax2.set_xlabel('Alanine Transaminase Level', {'font':'Serif', 'fontsize':16,'fontweight':'bold', 'color':'black'})

#-------------------------------------------------------
#axes titles
#ax1
fig.text(0.08, 0.65, " Overall ALT\nDistribution",
        {'font':'Serif', 'fontsize':20,'fontweight':'bold', 'color':'black'})
#ax2
fig.text(0.32, 0.68, "ALT Distribution\n by Liver Status",
        {'font':'Serif', 'fontsize':20,'fontweight':'bold', 'color':'black'})
#figure title
fig.text(0.05, 0.9, "High Alanine Transaminase Level May Indicate Liver Problem",
        {'font':'Serif', 'fontsize':25,'fontweight':'bold', 'color':'black'})

plt.show()
```



Feature : 'ALT'

Description : It indicates liver damage from hepatitis, infection, cirrhosis, liver cancer, or other liver diseases.

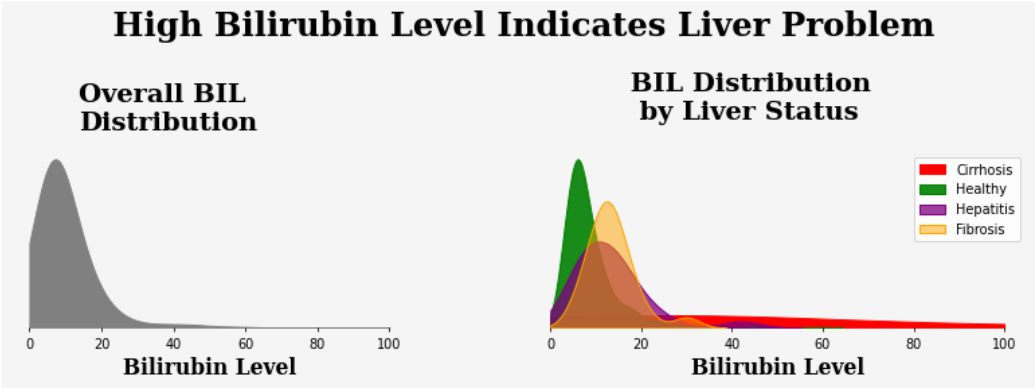
In [17]:
```
# plots univariate analysis of AST

fig = plt.figure(figsize = (24,4), dpi = 70)
gs = GridSpec(ncols=10, nrows=8, left=0.05, right=0.5, wspace=0.2, hspace=0.1)
fig.patch.set_facecolor('#f5f5f5')
sns.set_palette(sns.color_palette(['green','purple', 'orange', 'red']))

ax1 = fig.add_subplot(gs[3:8, 0:4])
ax2 = fig.add_subplot(gs[3:8, 5:])

# axes list
axes = [ ax1,ax2]

# setting of axes; visibility of axes and spines turn off
for ax in axes:

    ax.axes.get_yaxis().set_visible(False)
    #ax.axes.get_xaxis().set_visible(False)
    ax.set_facecolor('#f5f5f5')

    for loc in ['left', 'right', 'top', 'bottom']:
        ax.spines[loc].set_visible(False)

#-------Ax 1-----------------------------------------------
sns.kdeplot('AST', data=data, ax=ax1, shade=True, color='grey', alpha=1, clip=[0,400])
ax1.set_xlabel('Aspartate Aminotransferase Level', {'font':'Serif', 'fontsize':16,'fontweight':'bold', 'color':'black'

#-------Ax 2-----------------------------------------------
sns.kdeplot('AST', data=data[data.Category=='Cirrhosis'], ax=ax2, shade=True, color='red',
            alpha=1, label='Cirrhosis', clip=[0,400])
sns.kdeplot('AST', data=data[data.Category=='Healthy'], ax=ax2, shade=True, color='green',
            alpha=0.9, label='Healthy', clip=[0,400])
sns.kdeplot('AST', data=data[data.Category=='Hepatitis'], ax=ax2, shade=True, color='purple',
            alpha=0.75, label='Hepatitis', clip=[0,400])
sns.kdeplot('AST', data=data[data.Category=='Fibrosis'], ax=ax2, shade=True, color='orange',
            alpha=0.5, label='Fibrosis', clip=[0,400])

ax2.legend()
ax2.set_xlabel('Aspartate Aminotransferase Level', {'font':'Serif', 'fontsize':16,'fontweight':'bold', 'color':'black'

#------------------------------------------------------
#axes titles
#ax1
fig.text(0.08, 0.65, " Overall AST\nDistribution",
        {'font':'Serif', 'fontsize':15,'fontweight':'bold', 'color':'black'})
#ax2
fig.text(0.32, 0.68, "AST Distribution\n by Liver Status",
        {'font':'Serif', 'fontsize':15,'fontweight':'bold', 'color':'black'})

#figure title
fig.text(0.07, 0.9, "High Aspartate Aminotransferase Level Indicates Liver Problem",
        {'font':'Serif', 'fontsize':20,'fontweight':'bold', 'color':'black'})


plt.show()
```



Feature : 'AST'

Description : Aspartate aminotransferase is an enzyme that is found mostly in the liver, but also in muscles.

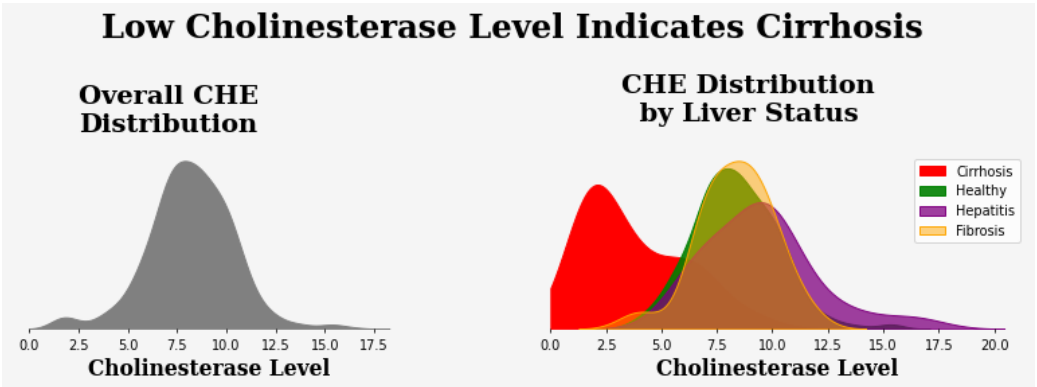
In [18]:
```
# plots univariate analysis of BIL

fig = plt.figure(figsize = (24,4), dpi = 70)
gs = GridSpec(ncols=10, nrows=8, left=0.05, right=0.5, wspace=0.2, hspace=0.1)
fig.patch.set_facecolor('#f5f5f5')
sns.set_palette(sns.color_palette(['green','purple', 'orange', 'red']))

ax1 = fig.add_subplot(gs[3:8, 0:4])
ax2 = fig.add_subplot(gs[3:8, 5:])

# axes list
axes = [ ax1,ax2]

# setting of axes; visibility of axes and spines turn off
for ax in axes:

    ax.axes.get_yaxis().set_visible(False)
    #ax.axes.get_xaxis().set_visible(False)
    ax.set_facecolor('#f5f5f5')

    for loc in ['left', 'right', 'top', 'bottom']:
        ax.spines[loc].set_visible(False)


#-------Ax 1--------------------------------------------------
sns.kdeplot('BIL', data=data, ax=ax1, shade=True, color='grey', alpha=1, clip=[0,100])
ax1.set_xlabel('Bilirubin Level', {'font':'Serif', 'fontsize':16,'fontweight':'bold', 'color':'black'})

#-------Ax 2--------------------------------------------------
sns.kdeplot('BIL', data=data[data.Category=='Cirrhosis'], ax=ax2, shade=True, color='red',
            alpha=1, label='Cirrhosis', clip=[0,100])
sns.kdeplot('BIL', data=data[data.Category=='Healthy'], ax=ax2, shade=True, color='green',
            alpha=0.9, label='Healthy', clip=[0,100])
sns.kdeplot('BIL', data=data[data.Category=='Hepatitis'], ax=ax2, shade=True, color='purple',
            alpha=0.75, label='Hepatitis', clip=[0,100])
sns.kdeplot('BIL', data=data[data.Category=='Fibrosis'], ax=ax2, shade=True, color='orange',
            alpha=0.5, label='Fibrosis', clip=[0,100])

ax2.legend()
ax2.set_xlabel('Bilirubin Level', {'font':'Serif', 'fontsize':16,'fontweight':'bold', 'color':'black'})

#-------------------------------------------------------------
#axes titles
#ax1
fig.text(0.08, 0.65, "Overall BIL\nDistribution",
        {'font':'Serif', 'fontsize':20,'fontweight':'bold', 'color':'black'})
#ax2
fig.text(0.32, 0.68, " BIL Distribution\n  by Liver Status",
        {'font':'Serif', 'fontsize':20,'fontweight':'bold', 'color':'black'})
#figure title
fig.text(0.08, 0.9, "   High Bilirubin Level Indicates Liver Problem",
        {'font':'Serif', 'fontsize':25,'fontweight':'bold', 'color':'black'})


plt.show()
```



Feature : 'BIL'

Description : A bilirubin test measures the amount of bilirubin in the blood.

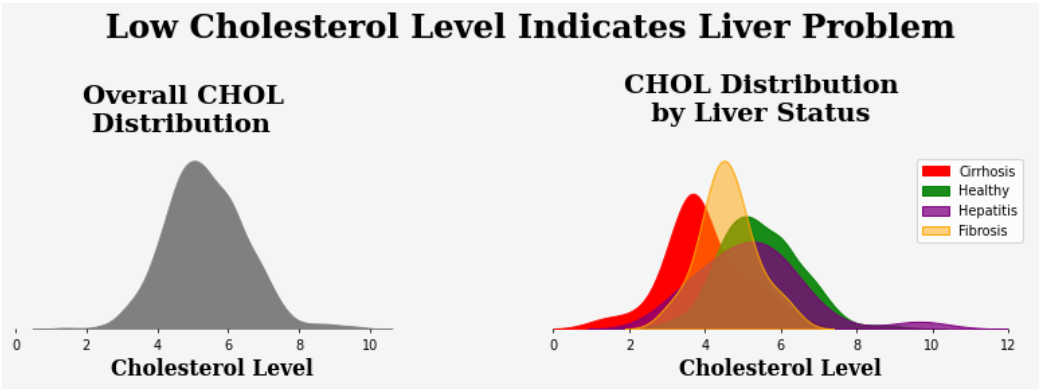
In [19]:
```
# plots univariate analysis of CHE

fig = plt.figure(figsize = (24,4), dpi = 70)
gs = GridSpec(ncols=10, nrows=8, left=0.05, right=0.5, wspace=0.2, hspace=0.1)
fig.patch.set_facecolor('#f5f5f5')
sns.set_palette(sns.color_palette(['green','purple', 'orange', 'red']))

ax1 = fig.add_subplot(gs[3:8, 0:4])
ax2 = fig.add_subplot(gs[3:8, 5:])

# axes list
axes = [ ax1,ax2]

# setting of axes; visibility of axes and spines turn off
for ax in axes:

    ax.axes.get_yaxis().set_visible(False)
    #ax.axes.get_xaxis().set_visible(False)
    ax.set_facecolor('#f5f5f5')

    for loc in ['left', 'right', 'top', 'bottom']:
        ax.spines[loc].set_visible(False)

#-------Ax 1-----------------------------------------------
sns.kdeplot('CHE', data=data, ax=ax1, shade=True, color='grey', alpha=1, clip=[0,400])
ax1.set_xlabel('Cholinesterase Level', {'font':'Serif', 'fontsize':16,'fontweight':'bold', 'color':'black'})

#-------Ax 2-----------------------------------------------
sns.kdeplot('CHE', data=data[data.Category=='Cirrhosis'], ax=ax2, shade=True, color='red',
            alpha=1, label='Cirrhosis', clip=[0,400])
sns.kdeplot('CHE', data=data[data.Category=='Healthy'], ax=ax2, shade=True, color='green',
            alpha=0.9, label='Healthy', clip=[0,400])
sns.kdeplot('CHE', data=data[data.Category=='Hepatitis'], ax=ax2, shade=True, color='purple',
            alpha=0.75, label='Hepatitis', clip=[0,400])
sns.kdeplot('CHE', data=data[data.Category=='Fibrosis'], ax=ax2, shade=True, color='orange',
            alpha=0.5, label='Fibrosis', clip=[0,400])

ax2.legend()
ax2.set_xlabel('Cholinesterase Level', {'font':'Serif', 'fontsize':16,'fontweight':'bold', 'color':'black'})

#-------------------------------------------------------
#axes titles
#ax1
fig.text(0.08, 0.65, "Overall CHE\nDistribution",
        {'font':'Serif', 'fontsize':20,'fontweight':'bold', 'color':'black'})
#ax2
fig.text(0.32, 0.68, "CHE Distribution\n  by Liver Status",
        {'font':'Serif', 'fontsize':20,'fontweight':'bold', 'color':'black'})
#figure title
fig.text(0.09, 0.9, "Low Cholinesterase Level Indicates Cirrhosis",
        {'font':'Serif', 'fontsize':25,'fontweight':'bold', 'color':'black'})

plt.show()
```



Feature : 'CHE'

Description : Serum cholinesterase (ChE) is an enzyme synthesized by hepatocytes and its levels reflect the synthetic function of the liver.

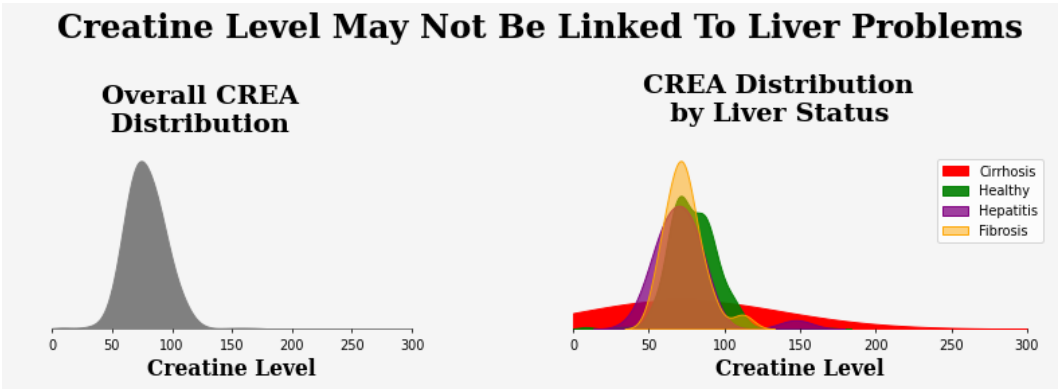
In [20]:
```
# plots univariate analysis of CHOL

fig = plt.figure(figsize = (24,4), dpi = 70)
gs = GridSpec(ncols=10, nrows=8, left=0.05, right=0.5, wspace=0.2, hspace=0.1)
fig.patch.set_facecolor('#f5f5f5')
sns.set_palette(sns.color_palette(['green','purple', 'orange', 'red']))

ax1 = fig.add_subplot(gs[3:8, 0:4])
ax2 = fig.add_subplot(gs[3:8, 5:])

# axes list
axes = [ ax1,ax2]

# setting of axes; visibility of axes and spines turn off
for ax in axes:

    ax.axes.get_yaxis().set_visible(False)
    #ax.axes.get_xaxis().set_visible(False)
    ax.set_facecolor('#f5f5f5')

    for loc in ['left', 'right', 'top', 'bottom']:
        ax.spines[loc].set_visible(False)

#-------Ax 1-----------------------------------------------
sns.kdeplot('CHOL', data=data, ax=ax1, shade=True, color='grey', alpha=1, clip=[0,400])
ax1.set_xlabel('Cholesterol Level', {'font':'Serif', 'fontsize':16,'fontweight':'bold', 'color':'black'})

#-------Ax 2-----------------------------------------------
sns.kdeplot('CHOL', data=data[data.Category=='Cirrhosis'], ax=ax2, shade=True, color='red',
            alpha=1, label='Cirrhosis', clip=[0,400])
sns.kdeplot('CHOL', data=data[data.Category=='Healthy'], ax=ax2, shade=True, color='green',
            alpha=0.9, label='Healthy', clip=[0,400])
sns.kdeplot('CHOL', data=data[data.Category=='Hepatitis'], ax=ax2, shade=True, color='purple',
            alpha=0.75, label='Hepatitis', clip=[0,400])
sns.kdeplot('CHOL', data=data[data.Category=='Fibrosis'], ax=ax2, shade=True, color='orange',
            alpha=0.5, label='Fibrosis', clip=[0,400])

ax2.legend()
ax2.set_xlabel('Cholesterol Level', {'font':'Serif', 'fontsize':16,'fontweight':'bold', 'color':'black'})

#-------------------------------------------------------
#axes titles
#ax1
fig.text(0.08, 0.65, "Overall CHOL\n Distribution",
        {'font':'Serif', 'fontsize':20,'fontweight':'bold', 'color':'black'})
#ax2
fig.text(0.32, 0.68, "CHOL Distribution\n   by Liver Status",
        {'font':'Serif', 'fontsize':20,'fontweight':'bold', 'color':'black'})
#figure title
fig.text(0.08, 0.9, "  Low Cholesterol Level Indicates Liver Problem",
        {'font':'Serif', 'fontsize':25,'fontweight':'bold', 'color':'black'})

plt.show()
```



Feature : 'CHOL'

Description : It can measure the amount of cholesterol and triglycerides in your blood

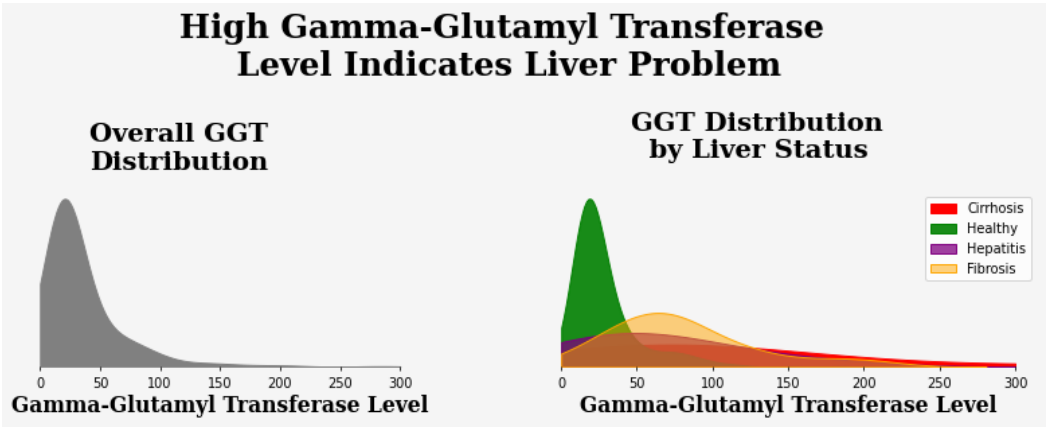
In [21]:
```
# plots univariate analysis of CREA

fig = plt.figure(figsize = (24,4), dpi = 70)
gs = GridSpec(ncols=10, nrows=8, left=0.05, right=0.5, wspace=0.2, hspace=0.1)
fig.patch.set_facecolor('#f5f5f5')
sns.set_palette(sns.color_palette(['green','purple', 'orange', 'red']))

ax1 = fig.add_subplot(gs[3:8, 0:4])
ax2 = fig.add_subplot(gs[3:8, 5:])

# axes list
axes = [ ax1,ax2]

# setting of axes; visibility of axes and spines turn off
for ax in axes:

    ax.axes.get_yaxis().set_visible(False)
    #ax.axes.get_xaxis().set_visible(False)
    ax.set_facecolor('#f5f5f5')

    for loc in ['left', 'right', 'top', 'bottom']:
        ax.spines[loc].set_visible(False)

#-------Ax 1--------------------------------------------------
sns.kdeplot('CREA', data=data, ax=ax1, shade=True, color='grey', alpha=1, clip=[0,300])
ax1.set_xlabel('Creatine Level', {'font':'Serif', 'fontsize':16,'fontweight':'bold', 'color':'black'})

#-------Ax 2--------------------------------------------------
sns.kdeplot('CREA', data=data[data.Category=='Cirrhosis'], ax=ax2, shade=True, color='red',
            alpha=1, label='Cirrhosis', clip=[0,300])
sns.kdeplot('CREA', data=data[data.Category=='Healthy'], ax=ax2, shade=True, color='green',
            alpha=0.9, label='Healthy', clip=[0,300])
sns.kdeplot('CREA', data=data[data.Category=='Hepatitis'], ax=ax2, shade=True, color='purple',
            alpha=0.75, label='Hepatitis', clip=[0,300])
sns.kdeplot('CREA', data=data[data.Category=='Fibrosis'], ax=ax2, shade=True, color='orange',
            alpha=0.5, label='Fibrosis', clip=[0,300])

ax2.legend()
ax2.set_xlabel('Creatine Level', {'font':'Serif', 'fontsize':16,'fontweight':'bold', 'color':'black'})

#-------------------------------------------------------------
#axes titles
#ax1
fig.text(0.08, 0.65, "Overall CREA\n Distribution",
        {'font':'Serif', 'fontsize':20,'fontweight':'bold', 'color':'black'})
#ax2
fig.text(0.32, 0.68, "CREA Distribution\n   by Liver Status",
        {'font':'Serif', 'fontsize':20,'fontweight':'bold', 'color':'black'})
#figure title
fig.text(0.04, 0.9, "    Creatine Level May Not Be Linked To Liver Problems",
        {'font':'Serif', 'fontsize':25,'fontweight':'bold', 'color':'black'})

plt.show()
```



Feature : 'CREA'

Description : A creatinine test is a measure of how well kidneys are performing their job of filtering waste from your blood.

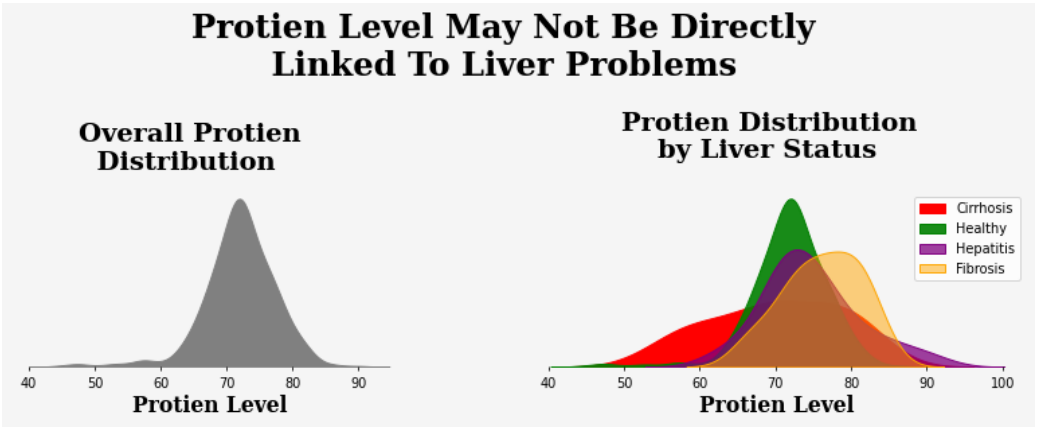
In [22]:
```
# plots univariate analysis of GGT

fig = plt.figure(figsize = (24,4), dpi = 70)
gs = GridSpec(ncols=10, nrows=8, left=0.05, right=0.5, wspace=0.2, hspace=0.1)
fig.patch.set_facecolor('#f5f5f5')
sns.set_palette(sns.color_palette(['green','purple', 'orange', 'red']))

ax1 = fig.add_subplot(gs[3:8, 0:4])
ax2 = fig.add_subplot(gs[3:8, 5:])

# axes list
axes = [ ax1,ax2]

# setting of axes; visibility of axes and spines turn off
for ax in axes:

    ax.axes.get_yaxis().set_visible(False)
    #ax.axes.get_xaxis().set_visible(False)
    ax.set_facecolor('#f5f5f5')

    for loc in ['left', 'right', 'top', 'bottom']:
        ax.spines[loc].set_visible(False)

#-------Ax 1-----------------------------------------------
sns.kdeplot('GGT', data=data, ax=ax1, shade=True, color='grey', alpha=1, clip=[0,300])
ax1.set_xlabel('Gamma-Glutamyl Transferase Level', {'font':'Serif', 'fontsize':16,'fontweight':'bold', 'color':'black'

#-------Ax 2-----------------------------------------------
sns.kdeplot('GGT', data=data[data.Category=='Cirrhosis'], ax=ax2, shade=True, color='red',
            alpha=1, label='Cirrhosis', clip=[0,300])
sns.kdeplot('GGT', data=data[data.Category=='Healthy'], ax=ax2, shade=True, color='green',
            alpha=0.9, label='Healthy', clip=[0,300])
sns.kdeplot('GGT', data=data[data.Category=='Hepatitis'], ax=ax2, shade=True, color='purple',
            alpha=0.75, label='Hepatitis', clip=[0,300])
sns.kdeplot('GGT', data=data[data.Category=='Fibrosis'], ax=ax2, shade=True, color='orange',
            alpha=0.5, label='Fibrosis', clip=[0,300])

ax2.legend()
ax2.set_xlabel('Gamma-Glutamyl Transferase Level', {'font':'Serif', 'fontsize':16,'fontweight':'bold', 'color':'black'

#-------------------------------------------------------
#axes titles
#ax1
fig.text(0.08, 0.65, "Overall GGT\nDistribution",
        {'font':'Serif', 'fontsize':20,'fontweight':'bold', 'color':'black'})
#ax2
fig.text(0.32, 0.68, "GGT Distribution\n  by Liver Status",
        {'font':'Serif', 'fontsize':20,'fontweight':'bold', 'color':'black'})
#figure title
fig.text(0.12, 0.9, "High Gamma-Glutamyl Transferase\n     Level Indicates Liver Problem",
        {'font':'Serif', 'fontsize':25,'fontweight':'bold', 'color':'black'})

plt.show()
```



Feature : 'GGT'

Description : An elevation of gamma-glutamyl transferase (GGT) activity is seen in many forms of liver disease.

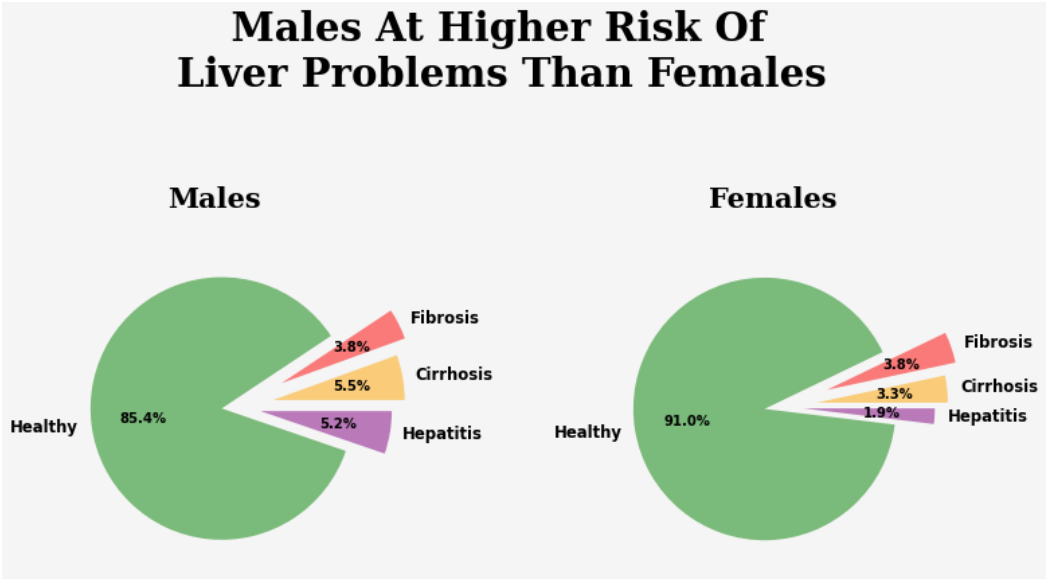
In [23]:
```
# plots univariate analysis of PROT

fig = plt.figure(figsize = (24,4), dpi = 70)
gs = GridSpec(ncols=10, nrows=8, left=0.05, right=0.5, wspace=0.2, hspace=0.1)
fig.patch.set_facecolor('#f5f5f5')
sns.set_palette(sns.color_palette(['green','purple', 'orange', 'red']))

ax1 = fig.add_subplot(gs[3:8, 0:4])
ax2 = fig.add_subplot(gs[3:8, 5:])

# axes list
axes = [ ax1,ax2]

# setting of axes; visibility of axes and spines turn off
for ax in axes:

    ax.axes.get_yaxis().set_visible(False)
    #ax.axes.get_xaxis().set_visible(False)
    ax.set_facecolor('#f5f5f5')

    for loc in ['left', 'right', 'top', 'bottom']:
        ax.spines[loc].set_visible(False)

#-------Ax 1-----------------------------------------------
sns.kdeplot('PROT', data=data, ax=ax1, shade=True, color='grey', alpha=1, clip=[0,300])
ax1.set_xlabel('Protien Level', {'font':'Serif', 'fontsize':16,'fontweight':'bold', 'color':'black'})

#-------Ax 2-----------------------------------------------
sns.kdeplot('PROT', data=data[data.Category=='Cirrhosis'], ax=ax2, shade=True, color='red',
            alpha=1, label='Cirrhosis', clip=[0,300])
sns.kdeplot('PROT', data=data[data.Category=='Healthy'], ax=ax2, shade=True, color='green',
            alpha=0.9, label='Healthy', clip=[0,300])
sns.kdeplot('PROT', data=data[data.Category=='Hepatitis'], ax=ax2, shade=True, color='purple',
            alpha=0.75, label='Hepatitis', clip=[0,300])
sns.kdeplot('PROT', data=data[data.Category=='Fibrosis'], ax=ax2, shade=True, color='orange',
            alpha=0.5, label='Fibrosis', clip=[0,300])

ax2.legend()
ax2.set_xlabel('Protien Level', {'font':'Serif', 'fontsize':16,'fontweight':'bold', 'color':'black'})

#-------------------------------------------------------
#axes titles
#ax1
fig.text(0.08, 0.65, "Overall Protien\n  Distribution",
        {'font':'Serif', 'fontsize':20,'fontweight':'bold', 'color':'black'})
#ax2
fig.text(0.32, 0.68, "Protien Distribution\n    by Liver Status",
        {'font':'Serif', 'fontsize':20,'fontweight':'bold', 'color':'black'})
#fig title
fig.text(0.13, 0.9, "Protien Level May Not Be Directly\n       Linked To Liver Problems",
        {'font':'Serif', 'fontsize':25,'fontweight':'bold', 'color':'black'})

plt.show()
```



Feature : 'PROT'

Description : A total protein test measures the amount of protein in your blood

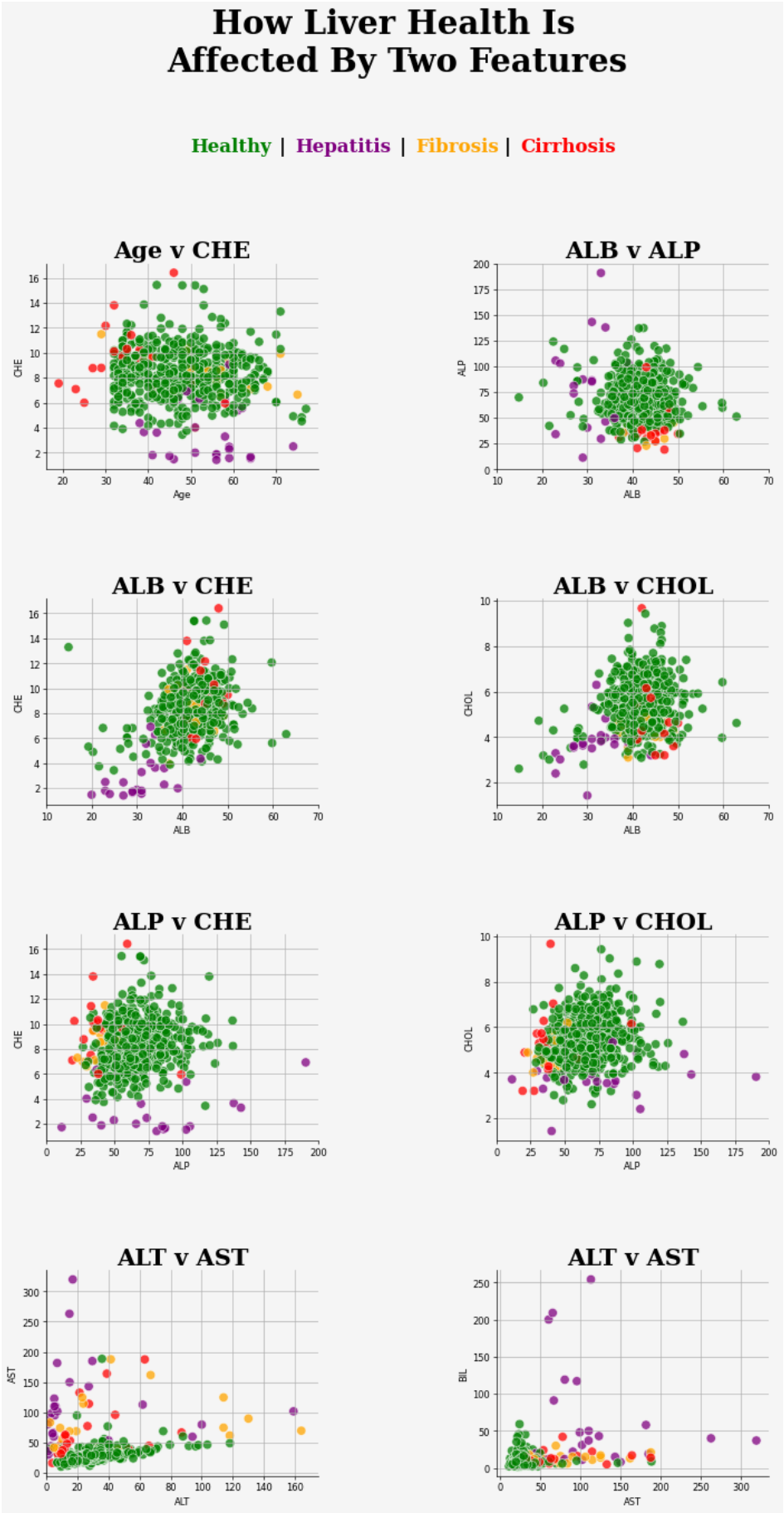
In [24]:
```
# plots univariate analysis of Sex

fig = plt.figure(figsize = (24,10), dpi = 60)
gs = GridSpec(ncols=13, nrows=5, left=0.05, right=0.5, wspace=0.2, hspace=0.1)
fig.patch.set_facecolor('#f5f5f5')
sns.set_palette(sns.color_palette(['green','purple', 'orange', 'red']))

ax1 = fig.add_subplot(gs[:, 0:5])
ax2 = fig.add_subplot(gs[:, 8:])


# axes list
axes = [ ax1,ax2]

# setting of axes; visibility of axes and spines turn off
for ax in axes:
    ax.axes.get_yaxis().set_visible(False)
    ax.set_facecolor('#f5f5f5')

    for loc in ['left', 'right', 'top', 'bottom']:
        ax.spines[loc].set_visible(False)

#-------------------------------------------------------------
#ax1
ax1_plot = ax1.pie(data[data.Sex == 'm'].groupby('Category').Category.count().values,
                   labels=data.groupby('Category').Category.count().index,
                    autopct='%1.1f%%', explode=[0.3,  0.4, 0.1, 0.2],
                   colors=['orange', 'red', 'green', 'purple'])

for piece in ax1_plot[0]:
    piece.set_alpha(0.5)

for i, text in enumerate(ax1_plot[1]):
    text.set_weight('bold')
    text.set_size(14)

for i, text in enumerate(ax1_plot[2]):
    text.set_weight('bold')
    text.set_size(12)


fig.text(0.1, 0.75, 'Males',  {'font':'Serif', 'weight':'bold','color': 'black', 'size':25})
#-------------------------------------------------------------
#ax2
ax2_plot = ax2.pie(data[data.Sex == 'f'].groupby('Category').Category.count().values,
                   labels=data.groupby('Category').Category.count().index,
                    autopct='%1.1f%%', explode=[0.3,  0.4, 0.1, 0.2],
                   colors=['orange', 'red', 'green', 'purple'])

for piece in ax2_plot[0]:
    piece.set_alpha(0.5)

for i, text in enumerate(ax2_plot[1]):
    text.set_weight('bold')
    text.set_size(14)

for i, text in enumerate(ax2_plot[2]):
    text.set_weight('bold')
    text.set_size(12)

fig.text(0.38, 0.75, 'Females',  {'font':'Serif', 'weight':'bold','color': 'black', 'size':25})
#--------------------------------
fig.text(0.105, 0.9, '    Males At Higher Risk Of\nLiver Problems Than Females',
         {'font':'Serif', 'weight':'bold','color': 'black', 'size':35})
plt.show()
```


Males At Higher Risk Of
Liver Problems Than Females
Males
Healthy
85.4%
Fibrosis
3.8%
Cirrhosis
5.5%
Hepatitis
5.2%
Females
Healthy
91.0%
Fibrosis
3.8%
Cirrhosis
3.3%
Hepatitis
1.9%

Bivariate Data Visualization:

In [25]:
```
# plots bivariate data

fig = plt.figure(figsize = (24,24), dpi = 60)
gs = GridSpec(ncols=13, nrows=29, left=0.05, right=0.5, wspace=0.2, hspace=0.1)
fig.patch.set_facecolor('#f5f5f5')
sns.set_palette(sns.color_palette(['green','purple', 'orange', 'red']))

ax1 = fig.add_subplot(gs[:5, :5])
ax2 = fig.add_subplot(gs[:5, 8:])
ax3 = fig.add_subplot(gs[8:13, :5])
ax4 = fig.add_subplot(gs[8:13, 8:])
ax5 = fig.add_subplot(gs[16:21, :5])
ax6 = fig.add_subplot(gs[16:21, 8:])
ax7 = fig.add_subplot(gs[24:, :5])
ax8 = fig.add_subplot(gs[24:, 8:])


# axes list
axes = [ ax1,ax2,ax3,ax4,ax5,ax6,ax7,ax8]

# setting of axes; visibility of axes and spines turn off
for ax in axes:
    ax.set_facecolor('#f5f5f5')
    ax.grid()

    for loc in ['right', 'top']:
        ax.spines[loc].set_visible(False)


#-------------------------------------------------------------------------------
#ax1 Age v CHE
sns.scatterplot(x='Age', y='CHE', hue='Category', data=data, ax=ax1, s=100, alpha=0.75, legend=None)
ax1.set_title('Age v CHE', {'font':'Serif', 'weight':'bold','color': 'black', 'size':25})
#-------------------------------------------------------------------------------
#ax2 ALB v ALP
sns.scatterplot(x='ALB', y='ALP', hue='Category', data=data, ax=ax2, s=100, alpha=0.75, legend=None)
ax2.set_title('ALB v ALP', {'font':'Serif', 'weight':'bold','color': 'black', 'size':25})
ax2.set_xlim(10, 70)
ax2.set_ylim(0,200)
#-------------------------------------------------------------------------------
#ax3 ALB v CHE
sns.scatterplot(x='ALB', y='CHE', hue='Category', data=data, ax=ax3, s=100, alpha=0.75, legend=None)
ax3.set_title('ALB v CHE', {'font':'Serif', 'weight':'bold','color': 'black', 'size':25})
ax3.set_xlim(10, 70)
#-------------------------------------------------------------------------------
#ax4 ALB v CHOL
sns.scatterplot(x='ALB', y='CHOL', hue='Category', data=data, ax=ax4, s=100, alpha=0.75, legend=None)
ax4.set_title('ALB v CHOL', {'font':'Serif', 'weight':'bold','color': 'black', 'size':25})
ax4.set_xlim(10, 70)
#-------------------------------------------------------------------------------
#ax5 ALP v CHE
sns.scatterplot(x='ALP', y='CHE', hue='Category', data=data, ax=ax5, s=100, alpha=0.75, legend=None)
ax5.set_title('ALP v CHE', {'font':'Serif', 'weight':'bold','color': 'black', 'size':25})
ax5.set_xlim(0, 200)
#-------------------------------------------------------------------------------
#ax6 ALP v CHOL
sns.scatterplot(x='ALP', y='CHOL', hue='Category', data=data, ax=ax6, s=100, alpha=0.75, legend=None)
ax6.set_title('ALP v CHOL', {'font':'Serif', 'weight':'bold','color': 'black', 'size':25})
ax6.set_xlim(0, 200)
#-------------------------------------------------------------------------------
#ax7 ALT v AST
sns.scatterplot(x='ALT', y='AST', hue='Category', data=data, ax=ax7, s=100, alpha=0.75, legend=None)
ax7.set_title('ALT v AST', {'font':'Serif', 'weight':'bold','color': 'black', 'size':25})
ax7.set_xlim(0, 175)
#-------------------------------------------------------------------------------
#ax8 AST v BIL
sns.scatterplot(x='AST', y='BIL', hue='Category', data=data, ax=ax8, s=100, alpha=0.75, legend=None)
ax8.set_title('ALT v AST', {'font':'Serif', 'weight':'bold','color': 'black', 'size':25})
#ax8.set_xlim(0, 175)


#-----------------------------------------
fig.text(0.1 + 0.04, 0.95, "Healthy",
         {'font':'Serif', 'weight':'bold','color': 'green', 'size':20})
fig.text(0.155 + 0.04, 0.95, "|",
         {'font':'Serif', 'weight':'bold','color': 'black', 'size':20})
fig.text(0.165 + 0.04, 0.95, "Hepatitis",
         {'font':'Serif', 'weight':'bold','color': 'purple', 'size':20})
fig.text(0.23 + 0.04, 0.95, "|",
         {'font':'Serif', 'weight':'bold','color': 'black', 'size':20})
fig.text(0.24 + 0.04, 0.95, "Fibrosis",
         {'font':'Serif', 'weight':'bold','color': 'orange', 'size':20})
fig.text(0.295 + 0.04, 0.95, "|",
         {'font':'Serif', 'weight':'bold','color': 'black', 'size':20})
fig.text(0.305 + 0.04, 0.95, "Cirrhosis",
```

```
        {'font':'Serif', 'weight':'bold','color': 'red', 'size':20})


fig.text(0.125, 1, "    How Liver Health Is\nAffected By Two Features",
        {'font':'Serif', 'weight':'bold','color': 'black', 'size':35})
plt.show()
```



The data collected from patients' records are not completely clear. Therefore, data cleaning is an essential step for developing machine learning models. Data preprocessing involves converting raw data into a logical or understandable format to ensure that the data have the same range of values and the features are comparable. Hence, the raw data were first normalized and converted into appropriate formats, which were more suitable for the different machine learning

estimators.

The missing values in our dataset were replaced with the mean value of each variable. Due to the presence of different measuring units, the data normalization method was performed using the StandardScaler function, which standardizes variables by scaling to unit variance.

In [67]:
```
# creates lower dimension dataset to visualize multiple features at once
# first preprocess data
# sets Sex to 0 (m) or 1 (f)

gender_map = {'m': 0, 'f': 1}
data['Sex'] = data.Sex.map(gender_map)

# creates pipeline for data processing
ct = ColumnTransformer([
    ('scaler', StandardScaler(), [0, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11])
], remainder='passthrough')

knn_imp = KNNImputer(n_neighbors=5)

pipeline = Pipeline([
    ('scale', ct),
    ('impute', knn_imp)
])

X = pipeline.fit_transform(data.iloc[:, 1:].values)
y = data.Category.values
```

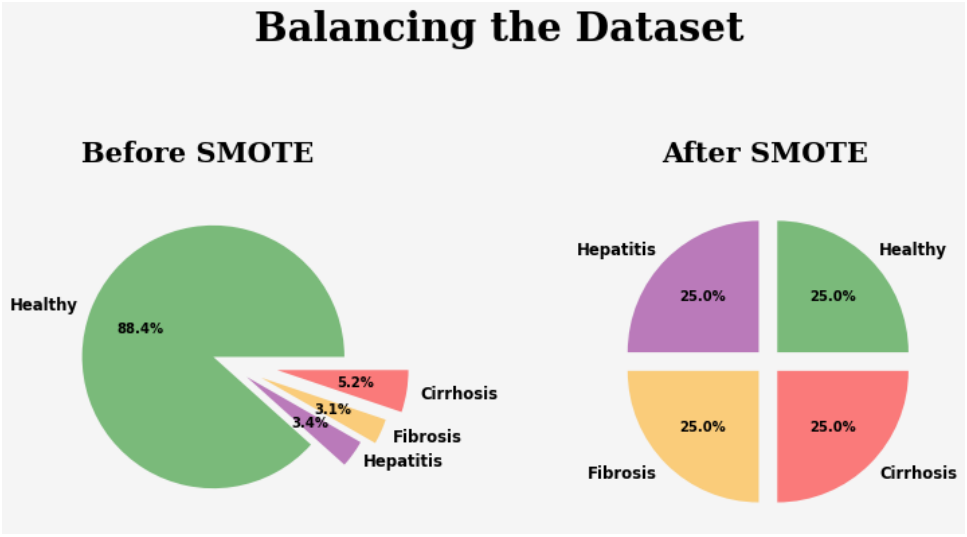
Converting Unbalanced Category Column to Balanced Using SMOTE :

The dataset is not balanced, which means that most of the records belonged to the same category. With Unbalanced data we can't train the model properly. Even if we train the model prediction will not be accurate.

Classification of imbalanced data is biased towards the large categories. The Symmetric Minority Over-sampling Technique (SMOTE) was applied to the HCV dataset to facilitate the performance of various classifiers. This method uses the KNN algorithm in creating new synthetic samples to balance the class distribution of the dataset.

In [68]:
```
# converts y to float values

y_float = np.where(y=='Healthy', 0.0, y)
y_float = np.where(y_float=='Hepatitis', 1.0, y_float)
y_float = np.where(y_float=='Fibrosis', 2.0, y_float)
y_float = np.where(y_float=='Cirrhosis', 3.0, y_float).astype('float64')
#splits data into train, validation, and test set
X_train, X_test, y_train, y_test = train_test_split(X, y_float, test_size=0.3, random_state=8)
#print(X_train)
#print(X_test)
X_val, X_test, y_val, y_test = train_test_split(X_test, y_test, test_size=0.5, random_state=8)
#SMOTE for class balancing
sm = SMOTE(random_state=8)

#create new training set with SMOTE object
X_bal, y_bal = sm.fit_resample(X_train, y_train)
```

In [69]:
```
# plots target data

fig = plt.figure(figsize = (24,10), dpi = 60)
gs = GridSpec(ncols=13, nrows=5, left=0.05, right=0.5, wspace=0.2, hspace=0.1)
fig.patch.set_facecolor('#f5f5f5')
sns.set_palette(sns.color_palette(['green','purple', 'orange', 'red']))

ax1 = fig.add_subplot(gs[:, 0:5])
ax2 = fig.add_subplot(gs[:, 8:])


# axes list
axes = [ ax1,ax2]

# setting of axes; visibility of axes and spines turn off
for ax in axes:
    ax.axes.get_yaxis().set_visible(False)
    ax.set_facecolor('#f5f5f5')

    for loc in ['left', 'right', 'top', 'bottom']:
        ax.spines[loc].set_visible(False)

#-------------------------------------------------------------
#ax1
pre_smote_count = [len(y_train[y_train==0.0]), len(y_train[y_train==1.0]),
                   len(y_train[y_train==2.0]), len(y_train[y_train==3.0])]
ax1_plot = ax1.pie(pre_smote_count,
                   labels=['Healthy', 'Hepatitis', 'Fibrosis', 'Cirrhosis'],
                    autopct='%1.1f%%', explode=[0.1,  0.2, 0.3, 0.4],
                   colors=['green', 'purple', 'orange', 'red'])

for piece in ax1_plot[0]:
    piece.set_alpha(0.5)

for i, text in enumerate(ax1_plot[1]):
    text.set_weight('bold')
    text.set_size(14)

for i, text in enumerate(ax1_plot[2]):
    text.set_weight('bold')
    text.set_size(12)


fig.text(0.06, 0.75, 'Before SMOTE',  {'font':'Serif', 'weight':'bold','color': 'black', 'size':25})
#-------------------------------------------------------------
#ax2
post_smote_count = [len(y_bal[y_bal==0.0]), len(y_bal[y_bal==1.0]),
                   len(y_bal[y_bal==2.0]), len(y_bal[y_bal==3.0])]
ax2_plot = ax2.pie(post_smote_count,
                   labels=['Healthy', 'Hepatitis', 'Fibrosis', 'Cirrhosis'],
                    autopct='%1.1f%%', explode=[0.1,  0.1, 0.1, 0.1],
                   colors=['green', 'purple', 'orange', 'red'])

for piece in ax2_plot[0]:
    piece.set_alpha(0.5)

for i, text in enumerate(ax2_plot[1]):
    text.set_weight('bold')
    text.set_size(14)

for i, text in enumerate(ax2_plot[2]):
    text.set_weight('bold')
    text.set_size(12)

fig.text(0.36, 0.75, 'After SMOTE',  {'font':'Serif', 'weight':'bold','color': 'black', 'size':25})
#---------------------------------
fig.text(0.15, 0.9, 'Balancing the Dataset',
         {'font':'Serif', 'weight':'bold','color': 'black', 'size':35})
plt.show()
```



We used OPTUNA technique for Hyperparameters optimization.

Read more of OPTUNA framework here https://optuna.readthedocs.io/en/stable/ (https://optuna.readthedocs.io/en/stable/)

In [70]:
```
# OPTUNA objective function

def objective(trial):
    #------------------------------
    #logistic regression
    lr_penalty = trial.suggest_categorical('lr_penalty', ['l1', 'l2', 'elasticnet'])
    lr_l1_ratio = None
    if lr_penalty == 'l1':
        lr_solver = trial.suggest_categorical('lr_solver1', ['liblinear', 'saga'])
    elif lr_penalty == 'l2':
        lr_solver = trial.suggest_categorical('lr_solver2', ['newton-cg', 'lbfgs', 'sag', 'liblinear', 'saga'])
    else:
        lr_solver = 'saga'
        lr_l1_ratio = trial.suggest_uniform('lr_l1_ratio', 0.0, 1.0)

    lr_tol = trial.suggest_uniform('lr_tol', 1e-5, 1e-2)
    lr_C = trial.suggest_uniform('lr_C', 0.0, 1.0)

    lr = LogisticRegression(
        penalty=lr_penalty,
        tol=lr_tol,
        C=lr_C,
        solver=lr_solver,
        l1_ratio=lr_l1_ratio
    )


    #------------------------------------------------------------
    #KNN
    knn_neighbors = trial.suggest_int('knn_neighbors', 2, 100)
    knn_weights = trial.suggest_categorical('knn_weights', ['uniform', 'distance'])
    knn_p = trial.suggest_categorical('knn_p', [1, 2])

    knn = KNeighborsClassifier(
        n_neighbors=knn_neighbors,
        weights=knn_weights,
        p=knn_p
    )


    #------------------------------------------------------------
    #SVM
    svm_C = trial.suggest_uniform('svm_C', 0.0, 1.0)
    svm_kernel = trial.suggest_categorical('svm_kernel', ['poly', 'rbf'])
    svm_degree = 3
    if svm_kernel == 'poly':
        svm_degree = trial.suggest_int('svm_degree', 1, 10)
    svm_tol = trial.suggest_uniform('svm_tol', 1e-5, 1e-2)

    svm = SVC(
        C=svm_C,
        kernel=svm_kernel,
        degree=svm_degree,
        tol=svm_tol
    )

    #------------------------------------------------------------
    #random forest
    rf_estimators = trial.suggest_int('rf_estimators', 1, 500)
    rf_criterion = trial.suggest_categorical('rf_criterion', ['entropy', 'gini'])
    rf_max_depth = trial.suggest_int('rf_max_depth', 1, 100)
    rf_min_samples_split = trial.suggest_int('rf_min_samples_split', 2, 50)
    rf_min_samples_leaf = trial.suggest_int('rf_min_samples_leaf', 1, 25)

    rf = RandomForestClassifier(
        n_estimators=rf_estimators,
        criterion=rf_criterion,
        max_depth=rf_max_depth,
        min_samples_split=rf_min_samples_split,
        min_samples_leaf=rf_min_samples_leaf
    )


    #------------------------------------------------------------
    #naive bayes
    nb_smoothing = trial.suggest_uniform('nb_smoothing', 1e-10, 1e-6)
    nb = GaussianNB(var_smoothing=nb_smoothing)


    #------------------------------------------------------------

    #ensemble model
    lr_w = trial.suggest_uniform('lr_w', 0.0, 1.0)
    knn_w = trial.suggest_uniform('knn_w', 0.0, 1.0)
    svm_w = trial.suggest_uniform('svm_w', 0.0, 1.0)
```

```
    rf_w = trial.suggest_uniform('rf_w', 0.0, 1.0)
    nb_w = trial.suggest_uniform('nb_w', 0.0, 1.0)


    vc = VotingClassifier(estimators=[
        ('lr', lr),
        ('knn', knn),
        ('svm', svm),
        ('rf', rf),
        ('nb', nb)],
                          weights=[lr_w, knn_w, svm_w, rf_w, nb_w]
                         )

    vc.fit(X_bal, y_bal)
    preds = vc.predict(X_val)

    acc = metrics.accuracy_score(y_val, preds)

    return acc
```

In [30]:
```
optuna.logging.set_verbosity(optuna.logging.ERROR)

study = optuna.create_study(direction='maximize')
study.optimize(objective, n_trials=100)
```

- Graphviz provides a simple pure-Python interface for the Graphviz graph-drawing software.
- This package is ude to create graph descriptions in the DOT language.
- More information(User Guide) is found here https://graphviz.readthedocs.io/en/stable/manual.html (https://graphviz.readthedocs.io/en/stable/manual.html)

In [31]:
```
# To better understand the process, draw the graph using 'graphviz' software
ensemble_graph = graphviz.Digraph()

#first layer
ensemble_graph.edge('data', 'Logistic Regression')
ensemble_graph.edge('data', 'KNN')
ensemble_graph.edge('data', 'SVM')
ensemble_graph.edge('data', 'Random Forest')
ensemble_graph.edge('data', 'Naive Bayes')

#second layer
ensemble_graph.edge('Logistic Regression', 'Ensemble Voter')
ensemble_graph.edge('KNN', 'Ensemble Voter')
ensemble_graph.edge('SVM', 'Ensemble Voter')
ensemble_graph.edge('Random Forest', 'Ensemble Voter')
ensemble_graph.edge('Naive Bayes', 'Ensemble Voter')

#third layer
ensemble_graph.edge('Ensemble Voter', 'prediction')

ensemble_graph
```

Out[31]:
```
<graphviz.graphs.Digraph at 0x12a70b520>
```

In [66]:
```
# print accuracy of the model and best hyperparameters

print('============================================================')
print('Model Accuracy on Valadation Set:', round(study.best_trial.values[0], 2))
print('============================================================')
print('Best Hyperparameters:')
print('============================================================')
print(study.best_params)
```

```
============================================================
Model Accuracy on Valadation Set: 0.93
============================================================
Best Hyperparameters:
============================================================
{'lr_penalty': 'l1', 'lr_solver1': 'saga', 'lr_tol': 0.0029213347496322337, 'lr_C': 0.4361331186523135, 'knn_neighbo
rs': 9, 'knn_weights': 'distance', 'knn_p': 1, 'svm_C': 0.9681409247676923, 'svm_kernel': 'poly', 'svm_degree': 3,
'svm_tol': 0.009317435567177425, 'rf_estimators': 17, 'rf_criterion': 'entropy', 'rf_max_depth': 23, 'rf_min_samples
_split': 37, 'rf_min_samples_leaf': 16, 'nb_smoothing': 1.4973829401315603e-07, 'lr_w': 0.0487128675886736, 'knn_w':
0.583758312031774, 'svm_w': 0.8174741007028012, 'rf_w': 0.35153391453272126, 'nb_w': 0.5842827528317647}
```

Model Creation based on VotingClassifier

A Voting Classifier is a machine learning model that trains on an ensemble of numerous models and predicts an output (class) based on their highest probability of chosen class as the output.

It simply aggregates the findings of each classifier passed into Voting Classifier and predicts the output class based on the highest majority of voting. The idea is instead of creating separate dedicated models and finding the accuracy for each them, we create a single model which trains by these models and predicts output based on their combined majority of voting for each output class.

References: https://www.geeksforgeeks.org/ml-voting-classifier-using-sklearn/ (https://www.geeksforgeeks.org/ml-voting-classifier-using-sklearn/) https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.VotingClassifier.html (https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.VotingClassifier.html)

In [32]:
```
#recreates a model from the best hyperparameters:

def create_model(best_params):
        #------------------------------
    try:
        l1_ratio = best_params['lr_l1_ratio']
    except:
        l1_ratio = None
    try:
        solver = best_params['lr_solver1']
    except:
        try:
            solver = best_params['lr_solver2']
        except:
            solver = 'saga'

    lr = LogisticRegression(
        penalty=best_params['lr_penalty'],
        tol=best_params['lr_tol'],
        C=best_params['lr_C'],
        l1_ratio=l1_ratio,
        solver=solver
    )


    #------------------------------------------------------------
    #KNN

    knn = KNeighborsClassifier(
        n_neighbors=best_params['knn_neighbors'],
        weights=best_params['knn_weights'],
        p=best_params['knn_p']
    )


    #------------------------------------------------------------
    #SVM
    try:
        svm_degree = best_params['svm_degree']
    except:
        svm_degree=3

    svm = SVC(
        C=best_params['svm_C'],
        kernel=best_params['svm_kernel'],
        degree=svm_degree,
        tol=best_params['svm_tol']
    )

    #------------------------------------------------------------
    #random forest
    rf = RandomForestClassifier(
        n_estimators=best_params['rf_estimators'],
        criterion=best_params['rf_criterion'],
        max_depth=best_params['rf_max_depth'],
        min_samples_split=best_params['rf_min_samples_split'],
        min_samples_leaf=best_params['rf_min_samples_leaf']
    )


    #------------------------------------------------------------
    #naive bayes
    nb = GaussianNB(var_smoothing=best_params['nb_smoothing'])


    #------------------------------------------------------------

    #ensemble model
    vc = VotingClassifier(estimators=[
        ('lr', lr),
        ('knn', knn),
        ('svm', svm),
        ('rf', rf),
        ('nb', nb)],
                          weights=[
                              best_params['lr_w'],
                              best_params['knn_w'],
                              best_params['svm_w'],
                              best_params['rf_w'],
                              best_params['nb_w']]
                         )

    vc.fit(X_bal, y_bal)
```

```
    return vc
```

In [34]:
```
#ensemble model with best hyperparameters
model = create_model(study.best_params)
```

In [35]:
```
print(model)
```

```
VotingClassifier(estimators=[('lr',
                              LogisticRegression(C=0.4361331186523135,
                                                 penalty='l1', solver='saga',
                                                 tol=0.0029213347496322337)),
                             ('knn',
                              KNeighborsClassifier(n_neighbors=9, p=1,
                                                   weights='distance')),
                             ('svm',
                              SVC(C=0.9681409247676923, kernel='poly',
                                  tol=0.009317435567177425)),
                             ('rf',
                              RandomForestClassifier(criterion='entropy',
                                                     max_depth=23,
                                                     min_samples_leaf=16,
                                                     min_samples_split=37,
                                                     n_estimators=17)),
                             ('nb',
                              GaussianNB(var_smoothing=1.4973829401315603e-07))],
                 weights=[0.0487128675886736, 0.583758312031774,
                          0.8174741007028012, 0.35153391453272126,
                          0.5842827528317647])
```

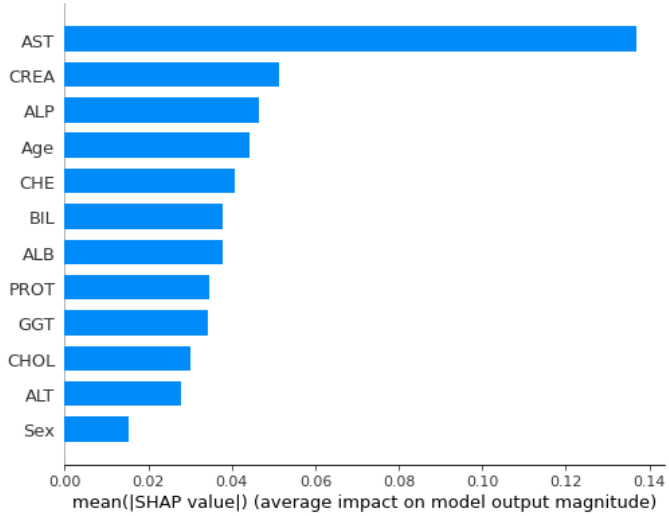
We used SHAP technique to know how each feature impacts the output.

SHAP (SHapley Additive exPlanations) is a game theoretic approach to explain the output of any machine learning model.

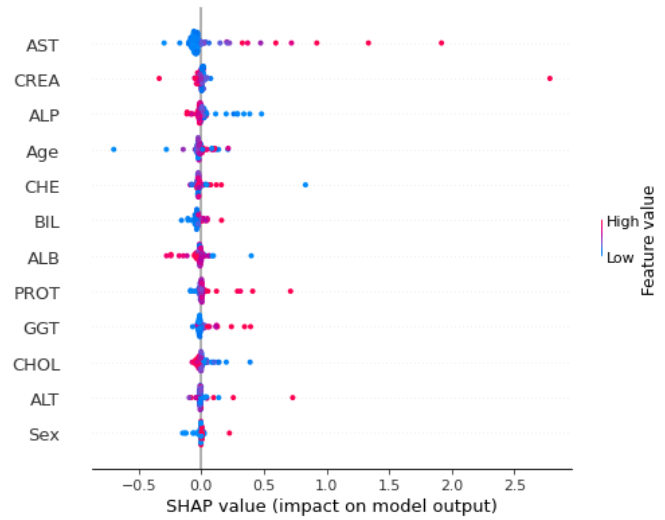
In [36]:
```
#creates shap explainer
feature_names = list(data.columns)[1:2] + list(data.columns)[3:] + [list(data.columns)[2]]
explainer = shap.Explainer(model.predict, X_train, feature_names=feature_names)
shap_values = explainer(X_test)
```

```
Permutation explainer: 84it [06:33,  4.74s/it]
```

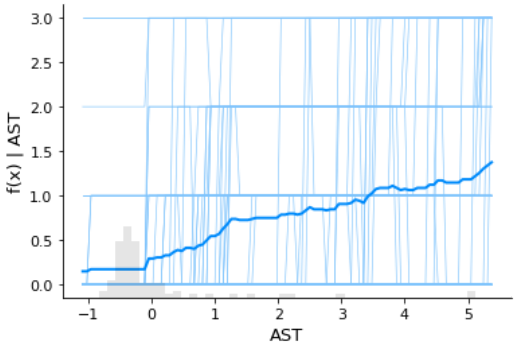
In [37]:
```
#plots importance of each feature
shap.summary_plot(shap_values, X_test, plot_type="bar")
```



In [38]:
```
#plots importance of each feature
shap.summary_plot(shap_values, X_test)
```



In [39]:
```
#plots effect of the AST feature
shap.partial_dependence_plot(4, model.predict, X_test, feature_names=feature_names)
```



The amount of aspartate aminotransferase in a patient's blood contributes significantly more to the ensemble model than any other feature.

From the above SHAP graphs, it is clearly seen that, the greater the AST level, the higher chance of being classified as having a liver disease.

In [41]:
```
#displays the final evaluation
test_preds = model.predict(X_test)
print('Test Accuracy:', round(metrics.accuracy_score(y_test, test_preds), 2))
```

```
Test Accuracy: 0.9
```

In [42]:
```
# print the predicted values
print(test_preds)
```

```
[0. 0. 0. 0. 0. 2. 0. 0. 1. 0. 0. 0. 0. 0. 2. 0. 0. 0. 0. 0. 0. 0. 3. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 1. 0. 0. 0.
 0. 3. 0. 1. 0. 0. 0. 0. 3. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 1. 1. 0. 0. 0. 0.]
```

Implementing the developed model to Test Data Set

In [ ]:
```

```

In [56]:
```
# load the test data
data_test = pd.read_csv('test_classification.csv')
```

In [57]:
```
# shape of the test data
data_test.shape
```

Out[57]:
```
(62, 12)
```

In [58]:
```
# looking at first 5 rows of test dataset
data_test.head()
```

Out[58]:

	Age	Sex	ALB	ALP	ALT	AST	BIL	CHE	CHOL	CREA	GGT	PROT
0	36	m	47.8	89.0	48.5	38.4	8.6	8.26	5.62	96.0	21.9	76.2
1	56	f	45.1	79.1	39.0	30.5	5.2	6.47	5.10	64.0	145.3	66.7
2	51	m	45.9	66.7	31.8	28.1	9.0	10.08	5.61	85.0	36.2	73.0
3	55	m	44.7	71.6	22.9	22.1	5.5	6.82	4.61	105.0	59.2	72.7
4	45	m	43.2	68.2	27.8	42.3	6.6	10.93	6.61	105.0	27.2	74.5

In [59]:
```
# converting categorical values to binary values
gender_map = {'m': 0, 'f': 1}
data_test['Sex'] = data_test.Sex.map(gender_map)
```

In [60]:
```
# check if there are any null values in the test dataset
print(data_test.isnull().sum())
```

```
Age     0
Sex     0
ALB     0
ALP     1
ALT     0
AST     0
BIL     0
CHE     0
CHOL    1
CREA    0
GGT     0
PROT    0
dtype: int64
```

In [61]:
```
# dropping the rows that contain null values
data_test = data_test.dropna(subset=['ALP','CHOL'])
data_test = data_test.reset_index(drop=True)
```

In [62]:
```
# verifying if there are any null values after droping them.
print(data_test.isnull().sum())
```

```
Age     0
Sex     0
ALB     0
ALP     0
ALT     0
AST     0
BIL     0
CHE     0
CHOL    0
CREA    0
GGT     0
PROT    0
dtype: int64
```

In [63]:
```
#creates pipeline for data processing

ct = ColumnTransformer([
    ('scaler', StandardScaler(), [0, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11])
], remainder='passthrough')

knn_imp = KNNImputer(n_neighbors=5)

pipeline = Pipeline([
    ('scale', ct),
    ('impute', knn_imp)
])

X_data_test = pipeline.fit_transform(data_test.iloc[:, :].values)
```

In [64]:
```
# giving new test data to the model for prediction
test_data_preds = model.predict(X_data_test)
```

In [65]:
```
# predicted values
print(test_data_preds)
```

```
[0. 0. 0. 0. 0. 3. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 3. 0. 0. 3.
 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 3. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 1. 0. 0. 0. 2. 0. 0. 0.]
```

Conclusion:

The final model has an accuracy of more than 90% on test data. The model is an voting ensemble of 5 models: logistic regression, k-nearest neighbors, support vector machine, random forest and naive bayes. Random forest and logistic regressing take up a majority of the vote.

In [ ]: