```
In [1]: # Clone Kaggle Jupyter Cheat Sheet repository
        !git clone https://github.com/ekhtiar/KaggleCheatSheet
```

Libraries

```
In [2]: import numpy as np
        import pandas as pd
        import seaborn as sns
        import matplotlib.pyplot as plt
        %matplotlib inline
        from sklearn.model_selection import train_test_split
        from sklearn.preprocessing import StandardScaler, MinMaxScaler, Normalizer
        from sklearn.metrics import confusion_matrix, accuracy_score
        from sklearn.tree import DecisionTreeClassifier
        from sklearn.ensemble import RandomForestClassifier
        from sklearn.svm import SVC, LinearSVC
        from sklearn.linear_model import LogisticRegression
        from sklearn.neighbors import KNeighborsClassifier
        from sklearn.naive_bayes import GaussianNB
        from sklearn.model_selection import cross_val_score, KFold, StratifiedKFold
        from sklearn.feature_selection import RFECV
        from sklearn.ensemble import ExtraTreesClassifier, GradientBoostingClassifier
        from sklearn.decomposition import PCA, KernelPCA
        from sklearn.model_selection import GridSearchCV, RandomizedSearchCV
        from sklearn.pipeline import Pipeline, FeatureUnion
        from scipy import stats, integrate
        import warnings
        warnings.filterwarnings('ignore')
        sns.set(color_codes=True)
```

Data Analysis

```
In [3]: df = pd.read_csv("KaggleCheatSheet/data_files/cleveland_heart_disease.csv")
        df.head()
```

	age	sexmale	chestpain_typeasymptomatic	chestpain_typeatypical angina	chestpain_typenon-anginal pain	chestpain_typetypical angina	restecg_normal	restecg_st-t abnormality	restecg_left ventricular hypertrophy
0	63.0	1	0.0	0.0	1.0	0.0	0.0	1.0	0.0
1	67.0	1	1.0	0.0	0.0	0.0	0.0	1.0	0.0
2	67.0	1	1.0	0.0	0.0	0.0	0.0	1.0	0.0
3	37.0	1	0.0	1.0	0.0	0.0	0.0	0.0	0.0
4	41.0	0	0.0	0.0	1.0	0.0	0.0	1.0	0.0

```
In [4]: df.describe()
```

	age	sexmale	chestpain_typeasymptomatic	chestpain_typeatypical angina	chestpain_typenon-anginal pain	chestpain_typetypical angina	restecg_normal	restecg_st-t abnormality	restecg_left ventricular hypertrophy
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000
mean	54.366337	0.683168	0.474257	0.120462	0.288779	0.166997	0.491749	0.144554	0.051485
std	9.082101	0.466011	0.499951	0.326051	0.453837	0.373068	0.500816	0.352474	0.221773
min	29.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	47.500000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
50%	55.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
75%	61.000000	1.000000	1.000000	0.000000	1.000000	0.000000	1.000000	0.000000	0.000000
max	77.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

```
In [5]: df.shape
```

```
Out[5]: (303, 14)
```

```
In [6]: corr = df.corr()
        sns.heatmap(corr,
                    xticklabels=corr.columns,
                    yticklabels=corr.columns,
                    annot=True,
                    cmap=sns.diverging_palette(220, 10, as_cmap=True))
```

```
In [7]: df['chestpain_typeasymptomatic'].hist(bins=50)
```

```
In [8]: sns.pairplot(df, x_vars=["age", "cholesterol", "resting_blood_pressure", "max_heart_rate_achieved"],
                     y_vars=["age", "cholesterol", "resting_blood_pressure", "max_heart_rate_achieved"],
                     hue="chestpain_typeasymptomatic", size=2.5, aspect=1.1, kde_kws={"bw": 1.5})
```

```
In [9]: df['age'].hist(bins=50)
```

```
In [10]: df['cholesterol'].hist(bins=50)
```

```
In [11]: df['resting_blood_pressure'].hist(bins=50)
```

```
In [12]: df['max_heart_rate_achieved'].hist(bins=50)
```

```
In [13]: df['fasting_blood_sugar_gt_120_mg/dl'].hist(bins=50)
```

```
In [14]: df['age'].hist(bins=50, label=['age'])
         df['cholesterol'].hist(bins=50, label=['cholesterol'])
         df['resting_blood_pressure'].hist(bins=50, label=['resting_blood_pressure'])
         df['max_heart_rate_achieved'].hist(bins=50, label=['max_heart_rate_achieved'])
         plt.legend(loc='upper right')
         plt.xlabel('Values')
         plt.ylabel('Frequency')
         plt.title('Histogram')
```

```
In [15]: df['age'].hist(bins=50, label=['age'])
         df['cholesterol'].hist(bins=50, label=['cholesterol'])
         df['resting_blood_pressure'].hist(bins=50, label=['resting_blood_pressure'])
         df['max_heart_rate_achieved'].hist(bins=50, label=['max_heart_rate_achieved'])
         plt.legend(loc='upper right')
         plt.xlabel('Values')
         plt.ylabel('Frequency')
         plt.title('Histogram')
         plt.xlim(0,500)
         plt.ylim(0,50)
```

```
In [16]: df.head()
```

	age	sexmale	chestpain_typeasymptomatic	chestpain_typeatypical angina	chestpain_typenon-anginal pain	chestpain_typetypical angina	restecg_normal	restecg_st-t abnormality	restecg_left ventricular hypertrophy
0	63.0	1	0.0	0.0	1.0	0.0	0.0	1.0	0.0
1	67.0	1	1.0	0.0	0.0	0.0	0.0	1.0	0.0
2	67.0	1	1.0	0.0	0.0	0.0	0.0	1.0	0.0
3	37.0	1	0.0	1.0	0.0	0.0	0.0	0.0	0.0
4	41.0	0	0.0	0.0	1.0	0.0	0.0	1.0	0.0

```
In [17]: df.groupby('sexmale').mean()
```

sexmale	age	chestpain_typeasymptomatic	chestpain_typeatypical angina	chestpain_typenon-anginal pain	chestpain_typetypical angina	restecg_normal	restecg_st-t abnormality	restecg_left ventricular hypertrophy
0	55.677083	0.791667	0.072917	0.239583	0.895833	0.104167	0.510417	0.156250
1	52.693431	0.374088	0.138686	0.303285	0.153285	0.182482	0.481752	0.138686

```
In [18]: corr = df.corr()
         sns.heatmap(corr,
                     xticklabels=corr.columns,
                     yticklabels=corr.columns,
                     annot=True,
                     cmap=sns.diverging_palette(220, 10, as_cmap=True))
```

```
In [19]: df.groupby('chestpain_typeasymptomatic').mean()
```

chestpain_typeasymptomatic	age	sexmale	chestpain_typeatypical angina	chestpain_typenon-anginal pain	chestpain_typetypical angina	restecg_normal	restecg_st-t abnormality	restecg_left ventricular hypertrophy
0.0	52.641026	0.641026	0.153846	0.358974	0.128205	0.487179	0.153846	0.051282
1.0	55.677083	0.791667	0.072917	0.239583	0.895833	0.104167	0.510417	0.156250

```
In [20]: corr = df.corr()
         sns.heatmap(corr,
                     xticklabels=corr.columns,
                     yticklabels=corr.columns,
                     annot=True,
                     cmap=sns.diverging_palette(220, 10, as_cmap=True))
```

```
In [21]: df.groupby('restecg_normal').mean()
```

restecg_normal	age	sexmale	chestpain_typeasymptomatic	chestpain_typeatypical angina	chestpain_typenon-anginal pain	chestpain_typetypical angina	restecg_st-t abnormality	restecg_left ventricular hypertrophy
0.0	54.420168	0.714286	0.487395	0.126050	0.294118	0.092437	0.168067	0.058824
1.0	54.293478	0.652174	0.461957	0.114130	0.282609	0.241304	0.000000	0.043478

```
In [22]: df.groupby('cholesterol').mean()
```

cholesterol	age	sexmale	chestpain_typeasymptomatic	chestpain_typeatypical angina	chestpain_typenon-anginal pain	chestpain_typetypical angina	restecg_normal	restecg_st-t abnormality	restecg_left ventricular hypertrophy
126	51.000000	1.000000	0.000000	0.000000	0.000000	1.000000	0.000000	0.000000	0.000000
131	55.000000	1.000000	0.000000	0.000000	0.000000	1.000000	0.000000	0.000000	0.000000
141	55.000000	0.000000	0.000000	0.000000	1.000000	0.000000	1.000000	0.000000	0.000000
149	53.000000	1.000000	0.000000	0.000000	0.000000	1.000000	0.000000	0.000000	0.000000
157	60.000000	1.000000	0.000000	0.000000	1.000000	0.000000	1.000000	0.000000	0.000000
164	38.000000	1.000000	0.000000	1.000000	0.000000	0.000000	1.000000	0.000000	0.000000
167	62.000000	0.000000	1.000000	0.000000	0.000000	0.000000	1.000000	0.000000	0.000000
168	53.000000	1.000000	0.000000	0.000000	1.000000	0.000000	1.000000	0.000000	0.000000
169	51.000000	1.000000	0.000000	0.000000	1.000000	0.000000	1.000000	0.000000	0.000000
170	60.000000	1.000000	0.000000	0.000000	1.000000	0.000000	1.000000	0.000000	0.000000

Feature Selection

```
In [23]: from sklearn.feature_selection import SelectKBest
         from sklearn.feature_selection import chi2, f_classif
         from sklearn.ensemble import ExtraTreesClassifier, RandomForestClassifier
         from sklearn.linear_model import LinearRegression, LogisticRegression, RandomizedLogisticRegression
         from sklearn.feature_selection import RFECV
         numpy_array = df.as_matrix()
         X = numpy_array[:,0:13]
         Y = numpy_array[:,13]
         print("Original shape: ", numpy_array.shape)
         print("Selected features: ", X.shape)
         print("Feature names: ", df.columns[0:13])
```

```
Original shape:  (303, 14)
Selected features:  (303, 13)
Feature names:  Index(['age', 'sexmale', 'chestpain_typeasymptomatic',
       'chestpain_typeatypical angina', 'chestpain_typenon-anginal pain',
       'chestpain_typetypical angina', 'restecg_normal',
       'restecg_st-t abnormality', 'restecg_left ventricular hypertrophy',
       'fasting_blood_sugar_gt_120_mg/dl', 'resting_blood_pressure',
       'cholesterol', 'max_heart_rate_achieved'],
      dtype='object')
```

```
In [24]: numpy_array[0:3,:]
```

```
Out[24]: array([ 63.,   1.,   0.,   0.,   1.,   0.,   0.,   1.,   0.,   1.,  145.,
         233.,  150.],
       [ 67.,   1.,   1.,   0.,   0.,   0.,   0.,   1.,   0.,   0.,  160.,
         286.,  108.],
       [ 67.,   1.,   1.,   0.,   0.,   0.,   0.,   1.,   0.,   0.,  120.,
         229.,  129.])
```

```
In [25]: from sklearn.feature_selection import VarianceThreshold
         from sklearn.preprocessing import StandardScaler
         X = numpy_array[:,0:13]
         Y = numpy_array[:,13]
         print("Original shape: ", numpy_array.shape)
         print("Selected features: ", X.shape)
         print("Feature names: ", df.columns[0:13])
         scaler = StandardScaler()
         rescaledX = scaler.fit_transform(X)
         print("Rescaled shape: ", rescaledX.shape)
         print("Rescaled features: ", df.columns[0:13])
```

```
Original shape:  (303, 14)
Selected features:  (303, 13)
Feature names:  Index(['age', 'sexmale', 'chestpain_typeasymptomatic',
       'chestpain_typeatypical angina', 'chestpain_typenon-anginal pain',
       'chestpain_typetypical angina', 'restecg_normal',
       'restecg_st-t abnormality', 'restecg_left ventricular hypertrophy',
       'fasting_blood_sugar_gt_120_mg/dl', 'resting_blood_pressure',
       'cholesterol', 'max_heart_rate_achieved'],
      dtype='object')
Rescaled shape:  (303, 13)
Rescaled features:  Index(['age', 'sexmale', 'chestpain_typeasymptomatic',
       'chestpain_typeatypical angina', 'chestpain_typenon-anginal pain',
       'chestpain_typetypical angina', 'restecg_normal',
       'restecg_st-t abnormality', 'restecg_left ventricular hypertrophy',
       'fasting_blood_sugar_gt_120_mg/dl', 'resting_blood_pressure',
       'cholesterol', 'max_heart_rate_achieved'],
      dtype='object')
```

```
In [26]: rescaledX[0:3,:]
```

```
Out[26]: array([ 0.94390,  0.68316,  0.09091,  0.07483,  0.21053,  0.12048,
         0.14286,  0.16129,  0.09091,  0.21053,  0.16667,  0.36986,
         0.22222],
       [ 1.39353,  0.68316,  0.90909,  0.07483,  0.00000,  0.00000,
         0.00000,  0.16129,  0.00000,  0.00000,  0.47826,  0.73059,
         0.16438],
       [ 1.39353,  0.68316,  0.90909,  0.07483,  0.00000,  0.00000,
         0.00000,  0.16129,  0.00000,  0.00000,  0.34783,  0.57078,
         0.19863],
       [ 0.94390,  0.68316,  0.09091,  0.91837,  0.00000,  0.00000,
         0.00000,  0.00000,  0.00000,  0.00000,  0.23188,  0.19635,
         0.30137],
       [ 0.35353,  0.68316,  0.09091,  0.07483,  0.36842,  0.00000,
         0.00000,  0.16129,  0.00000,  0.00000,  0.23188,  0.34247,
         0.24658],
       [ 1.35354,  0.68316,  0.09091,  0.07483,  0.00000,  0.00000,
         0.00000,  0.16129,  0.00000,  0.00000,  0.00000,  0.00000,
         0.00000],
       [ 1.35354,  0.68316,  0.09091,  0.07483,  0.00000,  0.00000,
         0.00000,  0.16129,  0.00000,  0.00000,  0.00000,  0.00000,
         0.00000],
       [ 0.35353,  0.68316,  0.09091,  0.07483,  0.00000,  0.00000,
         0.00000,  0.16129,  0.00000,  0.00000,  0.00000,  0.00000,
         0.00000],
       [ 1.35354
```

```
# Hyperparameter Regularization
# Define parameter grid for GradientBoosting
param_grid = {'n_estimators': [50, 100, 150],
              'learning_rate': [0.01, 0.1, 0.2],
              'max_depth': [3, 4, 5],
              'min_samples_split': [2, 5],
              'min_samples_leaf': [1, 2],
              'subsample': [0.8, 1.0]}
# Initialize GridSearchCV
grid_search = GridSearchCV(
    GradientBoostingClassifier(random_state=42),
    param_grid,
    cv=5,
    scoring='f1',
    verbose=1,
    n_jobs=-1
)
# Fit the model
print("Starting hyperparameter tuning...")
grid_search.fit(X_train, y_train)
# Get best parameters
print("\nBest parameters:")
for param, value in grid_search.best_params_.items():
    print(f"{param}: {value}")
# Get best model
best_model = grid_search.best_estimator_
# Evaluate on test set
y_pred = best_model.predict(X_test)
y_pred_proba = best_model.predict_proba(X_test)[:, 1]
print("\nTest set performance:")
print("Accuracy: {accuracy_score(y_test, y_pred):.4f}")
print("Precision: {precision_score(y_test, y_pred):.4f}")
print("Recall: {recall_score(y_test, y_pred):.4f}")
print("F1-score: {f1_score(y_test, y_pred):.4f}")
print("ROC-AUC: {roc_auc_score(y_test, y_pred_proba):.4f}")
```

```
Best hyperparameters for logistic Regression 0 optimizing f1 score 0.9440 0 0mean
```

Model Comparison

```
# Compare models
print("Comparison of all models:")
models = ['Random Forest', 'Gradient Boosting', 'Logistic Regression']
f1_scores = [0.9434, 0.9440, 0.9410]
roc_auc_scores = [0.9876, 0.9882, 0.9850]
for model, f1, roc_auc in zip(models, f1_scores, roc_auc_scores):
    print(f"{model}: F1={f1:.4f}, ROC-AUC={roc_auc:.4f}")
```

```
Comparison of all models:
Random Forest: F1=0.9434, ROC-AUC=0.9876
Gradient Boosting: F1=0.9440, ROC-AUC=0.9882
Logistic Regression: F1=0.9410, ROC-AUC=0.9850
```

```
# Visualization
plt.figure(figure_size=(8, 6))
models = ['Random Forest', 'Gradient Boosting', 'Logistic Regression']
f1_scores = [0.9434, 0.9440, 0.9410]
bars = plt.bar(models, f1_scores, color=['lightblue', 'lightgreen', 'lightcoral'])
plt.ylabel('F1 Score')
plt.title('Model Comparison')
plt.ylim(0.9400, 0.9450)
plt.xticks(rotation=45)
for bar, score in zip(bars, f1_scores):
    plt.text(bar.get_height(), bar.get_height() + 0.0005, f'{score:.4f}', ha='center')
plt.tight_layout()
plt.show()
```

Predictions

```
# Make predictions
print("Making predictions...")
predictions = best_model.predict(X_test)
probabilities = best_model.predict_proba(X_test)
print("Predictions made!")
print("Prediction shape: {predictions.shape}")
print("Probability shape: {probabilities.shape}")
```

```
Making predictions...
Predictions made!
Prediction shape: (2000,)
Probability shape: (2000, 2)
```

```
# Prediction distribution
print("Prediction distribution:")
unique, counts = np.unique(predictions, return_counts=True)
for cls, count in zip(unique, counts):
    print("Class {cls}: {count} ({count/len(predictions)*100:.1f}%)")
```

```
Prediction distribution:
Class 0: 1000 (50.0%)
Class 1: 1000 (50.0%)
```

```
# Confusion matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, predictions)
print("Confusion Matrix:")
print(cm)
```

```
Confusion Matrix:
[[ 971  29]
[  57 943]]
```

Confusion Matrix

```
# Visualize confusion matrix
plt.figure(figure_size=(6, 5))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.ylabel('True Label')
plt.xlabel('Predicted Label')
plt.title('Confusion Matrix')
plt.tight_layout()
plt.show()
```