```
#%%
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import matplotlib
from sklearn.model_selection import
train_test_split
from sklearn.preprocessing import
FunctionTransformer
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Ridge
from sklearn.feature_selection import RFE
from sklearn.linear_model import RidgeCV
from pandas import DataFrame
import matplotlib.pyplot as plt
import seaborn as sns
from pandas.plotting import scatter_matrix
from pandas import set_option
from pandas import read_csv
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import Normalizer
import math
import matplotlib.pyplot as plt
import numpy as np
from sklearn.linear_model import Lasso
from sklearn.linear_model import LassoCV
from numpy import set_printoptions
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.tree import DecisionTreeRegressor
from sklearn import metrics
#original (have to downgrade sklearns package to
use): from sklearn.externals.six import StringIO
from six import StringIO
from IPython.display import Image
from sklearn.tree import export_graphviz
#import pydotplus
from sklearn import tree
#import graphviz

from sklearn.ensemble import BaggingClassifier
```

```
39 from sklearn.tree import DecisionTreeClassifier
40 from sklearn.metrics import accuracy_score
41 from sklearn.ensemble import RandomForestClassifier
42 from sklearn.metrics import roc_curve
43 from sklearn.ensemble import AdaBoostClassifier
44
45 import warnings
46
47 from sklearn.kernel_ridge import KernelRidge
48
49 warnings.filterwarnings("ignore")
50 import pandas as pd
51 import numpy as np
52 import matplotlib.pyplot as plt
53 from sklearn.model_selection import KFold,
   cross_val_score
54 from sklearn.svm import SVC
55 from sklearn.svm import LinearSVC
56 from sklearn.metrics import accuracy_score
57 from sklearn.ensemble import RandomForestClassifier
   , AdaBoostClassifier
58 from sklearn.tree import DecisionTreeClassifier
59 from sklearn.metrics import accuracy_score,
   roc_auc_score, roc_curve
60 from sklearn.preprocessing import StandardScaler
61 from sklearn.model_selection import
   train_test_split
62 from sklearn.preprocessing import LabelEncoder
63 from sklearn.neural_network import MLPClassifier
64 from sklearn.datasets import make_classification
65 from sklearn.metrics import roc_auc_score,
   roc_curve, auc
66 #%%
67 file_name = 'Titanic train.csv'
68 df = read_csv(file_name)
69 #%%
70 def load_dataset(file_name):
71     """
72     Loads a dataset from a given file path.
73     """
74     return pd.read_csv(file_name)
```

```
 75
 76 def find_text_columns(df):
 77     """
 78     Identifies text columns in a DataFrame.
 79     Returns a list of column names that contain
    text.
 80     """
 81     # This is a simple heuristic; you might need a
     more sophisticated approach depending on your
    data
 82     text_columns = [col for col in df.columns if
    df[col].dtype == object]
 83     return text_columns
 84
 85 def convert_categorical_to_numeric(df, columns):
 86     """
 87     Converts specified categorical columns in a
    DataFrame to numerical values.
 88     Uses LabelEncoder to transform each unique
    category in the columns to a number.
 89     """
 90     le = LabelEncoder()
 91     for col in columns:
 92         df[col] = le.fit_transform(df[col])
 93     return df
 94
 95 # Columns to convert to numeric
 96 columns_to_convert = ['Sex', 'Ticket', 'Fare', '
    Cabin']
 97
 98 # Convert specified columns to numeric
 99 df_numerical = convert_categorical_to_numeric(df,
    columns_to_convert)
100
101 # Display the first few rows of the DataFrame
    after conversion
102 print(df_numerical.head())
103
104 df_numerical
105 #%%
106 df = df_numerical
```

```
107 df
108 #%%
109 df.drop("Name", axis=1, inplace=True)
110 df.drop("Embarked", axis=1, inplace=True)
111 df.drop("PassengerId", axis=1, inplace=True)
112 df = df.dropna()
113 df
114 #%%
115 X = df.drop(df.columns[0], axis = 1)
116 Y = df['Survived']
117 #%%
118 dataFrame = pd.DataFrame(X)
119 dataFrame
120 #%%
121 dataFrame.describe()
122 #%%
123 dataFrame.hist()
124 #%%
125 correlation_matrix = dataFrame.corr(method='
    pearson')
126 plt.figure(figsize=(10,8))
127 sns.heatmap(correlation_matrix, annot=True, cmap='
    coolwarm')
128 plt.show()
129 #%%
130 scatter_matrix(frame=dataFrame, alpha=0.5, figsize
    =(10,10), diagonal="hist")
131 scatter_matrix()
132 #%%
133 #Pre-processing
134 #Testing if normalization works better on the data
     than standardization
135 scaler1 = Normalizer().fit(dataFrame)
136 normalizedX = scaler1.transform(dataFrame)
137 dataNormDf = pd.DataFrame(normalizedX)
138 #%%
139 dataNormDf.describe()
140 #%%
141 dataNormDf.hist()
142 #%%
143 correlation_matrix = dataNormDf.corr(method='
```

```
pearson')
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.show()
#%%
#Pre-processing Standardization
#Testing if standardization works better on the data than normalization
scaler2 = StandardScaler().fit(X)
standardizedX = scaler2.transform(X)
dataStandDf = pd.DataFrame(standardizedX)
#%%
dataStandDf.describe()
#%%
dataStandDf.hist()
#%%
correlation_matrix = dataStandDf.corr(method='pearson')
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.show()
#%%
X_train, X_test, y_train, y_test = train_test_split(normalizedX, Y, test_size=0.20)
#%%
linear_svc = LinearSVC(C=100, loss = 'hinge', random_state=1, max_iter=1000000)
linear_svc.fit(X_train, y_train)
y_linear_pred = linear_svc.predict(X_test)

# determine accuracy score for the linear svc method
print("Linear SVC Method: " + str(accuracy_score(y_test, y_linear_pred)))
#%%
kernel_svc = SVC(kernel = 'rbf', degree = 2, C=1.0, random_state=1, max_iter = 1000000)
kernel_svc.fit(X_train, y_train)
y_kernel_pred = kernel_svc.predict(X_test)
```

```
175
176 # determine accuracy score for the linear svc
    method
177 print("Kernel SVC Method: " + str(accuracy_score(
    y_test, y_kernel_pred)))
178 #%%
179 random_forest_clf = RandomForestClassifier(
    random_state=42)
180 random_forest_clf.fit(X_train, y_train)
181 # y_prob_rf = random_forest_clf.predict_proba(
    X_test)
182 # y_pred_rf = random_forest_clf.predict(X_test)
183 # y_score_rf = y_prob_rf[:, 1]
184 # fpr_rf, tpr_rf, threshold_rf = roc_curve(y_test
    , y_score_rf)
185 y_rfclf_pred = random_forest_clf.predict(X_test)
186
187 # determine accuracy score for the linear svc
    method
188 print("Random Forest Classifier: " + str(
    accuracy_score(y_test, y_rfclf_pred)))
189 #%%
190 ada_clf = AdaBoostClassifier(n_estimators=200,
    algorithm= "SAMME.R", learning_rate=0.5,
    random_state=42)
191 ada_clf.fit(X_train, y_train)
192 y_ada_pred = ada_clf.predict(X_test)
193
194 # determine accuracy score for the linear svc
    method
195 print("Ada Boosting Method: " + str(accuracy_score
    (y_test, y_ada_pred)))
196 #%%
197 bag_clf = BaggingClassifier(
198     DecisionTreeClassifier(random_state=42),
    n_estimators=500,
199     max_samples=100, bootstrap=True, n_jobs=-1,
    random_state=42)
200 bag_clf.fit(X_train, y_train)
201 y_pred = bag_clf.predict(X_test)
202
```

```
# determine accuracy score for the bagging method
print("Bagging Method: " + str(accuracy_score(y_test, y_pred)))
#%%
# now use a standard decision tree classifier
tree_clf = DecisionTreeClassifier(random_state=42)
tree_clf.fit(X_train, y_train)
y_pred_tree = tree_clf.predict(X_test)

print("Standard Decision Tree Classifier: " + str(accuracy_score(y_test, y_pred_tree)))
#%%
clf = MLPClassifier(random_state=1, max_iter=300).fit(X_train, y_train)
clf.predict_proba(X_test[:1])

clf.predict(X_test[:5, :])

print("MLP Classifier: " + str(clf.score(X_test, y_test)))
#%%
# use cross-validation. Although we are building a single classification model
# prepare models
models = []
models.append(('Linear SVC', linear_svc))
models.append(('Kernel SVC', kernel_svc))
models.append(('Random Forest', random_forest_clf))
models.append(('AdaBoost', ada_clf))
models.append(('Bagging', bag_clf))
models.append(('Standard Decision Tree', tree_clf))
models.append(('MLP Classifier', clf))
#%%
# evaluate each model in turn
results = []
names = []
scoring = 'accuracy'

for name, model in models:
```

```
237     kfold = KFold(n_splits=10, random_state=7,
    shuffle = True)
238     cv_results = cross_val_score(model,
    normalizedX, Y, cv=kfold, scoring=scoring)
239     results.append(cv_results)
240     names.append(name)
241     msg = "%s: %f (%f)" % (name, cv_results.mean
    (), cv_results.std())
242     print(msg)
243
244 ## boxplot algorithm comparison
245 fig = plt.figure(figsize=(15,6))
246 fig.suptitle('Algorithm Comparison')
247 ax = fig.add_subplot(111)
248 plt.boxplot(results)
249 ax.set_xticklabels(names)
250 plt.show()
251 #%%
252 from sklearn.metrics import roc_curve, auc
253
254 def plot_roc_curve(classifier, X_test, y_test,
    title):
255     """
256     Plots the ROC curve for a given classifier.
257
258     Parameters:
259     - classifier: The trained classifier.
260     - X_test: Test data.
261     - y_test: True labels for the test data.
262     - title: Title for the ROC curve plot.
263     """
264     # Compute probabilities
265     y_probs = classifier.predict_proba(X_test)
266
267     # Compute ROC curve and ROC area for each
    class
268     fpr, tpr, _ = roc_curve(y_test, y_probs[:, 1])
269     roc_auc = auc(fpr, tpr)
270
271     plt.figure()
272     plt.plot(fpr, tpr, color='darkorange',
```

```
273                 lw=2, label=f'ROC curve (area = {
    roc_auc:.2f})')
274     plt.plot([0, 1], [0, 1], color='navy',
    linestyle='--')
275     plt.xlim([0.0, 1.0])
276     plt.ylim([0.0, 1.05])
277     plt.xlabel('False Positive Rate')
278     plt.ylabel('True Positive Rate')
279     plt.title(title)
280     plt.legend(loc="lower right")
281     plt.show()
282 #%%
283 # Usage for Random Forest Classifier
284 plot_roc_curve(random_forest_clf, X_test, y_test,
    'ROC Curve for Random Forest CLF')
285 #%%
286 # For AdaBoost ClF
287 plot_roc_curve(ada_clf, X_test, y_test, 'ROC Curve
     for AdaBoost CLF')
288 #%%
289 # For Bagging CLF
290 plot_roc_curve(bag_clf, X_test, y_test, 'ROC Curve
     for Bagging CLF')
291 #%%
292 # For Standard Decision Tree CLF
293 plot_roc_curve(tree_clf, X_test, y_test, 'ROC
    Curve for Standard Decision Tree CLF')
294 #%%
295 # For MLP Classifier
296 plot_roc_curve(clf, X_test, y_test, 'ROC Curve for
     MLP CLF')
297 #%%
298 # Load the new dataset
299 new_file = 'Titanic ML.csv'
300 data = read_csv(new_file)
301 #%%
302 def find_text_columns(data):
303     """
304     Identifies text columns in a DataFrame.
305     Returns a list of column names that contain
    text.
```

```
306     """
307     # This is a simple heuristic; you might need a more sophisticated approach depending on your data
308     text_columns = [col for col in data.columns if data[col].dtype == object]
309     return text_columns
310 
311 
312 def convert_categorical_to_numeric(data, columns):
313     """
314     Converts specified categorical columns in a DataFrame to numerical values.
315     Uses LabelEncoder to transform each unique category in the columns to a number.
316     """
317     le = LabelEncoder()
318     for col in columns:
319         data[col] = le.fit_transform(data[col])
320     return data
321 
322 
323 # Columns to convert to numeric
324 columns_to_convert = ['Sex', 'Ticket', 'Fare', 'Cabin']
325 
326 # Convert specified columns to numeric
327 data_numerical = convert_categorical_to_numeric(data, columns_to_convert)
328 #%%
329 # Display the first few rows of the DataFrame after conversion
330 print(data_numerical.head())
331 #%%
332 data_numerical
333 #%%
334 data = data_numerical
335 data
336 #%%
337 data.drop("Name", axis=1, inplace=True)
338 data.drop("Embarked", axis=1, inplace=True)
```

```
data = data.dropna()
data1 = data.select_dtypes(exclude=['object'])
data.drop("PassengerId", axis=1, inplace=True)
data1
#%%
random_forest_clf.predict(data)
#%%
list(random_forest_clf.predict(data))
#%%
pd.DataFrame({'PassengerID':data1.pop('PassengerId
') , 'Survived':random_forest_clf.predict(data
) }).to_csv('passenger_prediction.csv',index=False
)
```