```
# imports
import pandas as pd #useful for loading the dataset
import numpy as np #to perform array
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression, LogisticRegression
from sklearn.metrics import mean_squared_error, r2_score, roc_curve, roc_auc_score
import pickle
from imblearn.over_sampling import SMOTE
```

```
# download the files from google drive
from google.colab import files
uploaded = files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving Final_ProjectData.xlsx to Final_ProjectData.xlsx
```

```
# read the file
dataset = pd.read_excel('Final_ProjectData.xlsx', sheet_name='Sheet1', header=1)
```

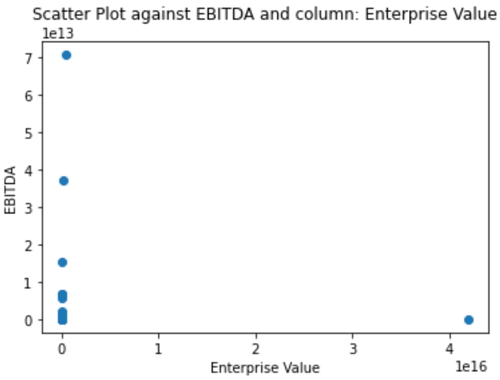
```
# inspect the dataset
print(dataset.shape)
print(dataset.head(5))
print(dataset.columns)
```

```
(1577, 11)
  Companies    MScore Fraud or Non Fraud        EBITDA Rating  R_Score  \
0       XOM -2.389523          Non Fraud  5.223000e+10     A+        5
1       JNJ -3.305048          Non Fraud  3.034900e+10     A+        5
2         V -3.262427          Non Fraud  1.953500e+10     A+        5
3       WMT -3.321651          Non Fraud  3.108100e+10     S-        5
4       CVX -3.039177          Non Fraud  4.021200e+10     A+        4

   Enterprise Value     MarketCap  Rev_Growth  NI_Growth  Debt Growth
0      3.626702e+11  3.217682e+11    0.549453   2.026738    -0.276774
1      4.690373e+11  4.497733e+11    0.135511   0.418921    -0.042959
2      4.130307e+11  4.062697e+11    0.215930   0.214930     0.070220
3      4.227217e+11  3.801587e+11    0.024328   0.012065    -0.093650
4      2.699732e+11  2.442442e+11    0.647130   3.818871    -0.292136
Index(['Companies', 'MScore', 'Fraud or Non Fraud', 'EBITDA', 'Rating',
       'R_Score', 'Enterprise Value', 'MarketCap', 'Rev_Growth', 'NI_Growth',
       'Debt Growth'],
      dtype='object')
```

```
def visualizeBar(df, column):
    x = df[column]
    y = df['EBITDA']

    # Create a scatter plot
    plt.scatter(x, y)

    # set the chart title and axis labels
    plt.title('Scatter Plot against EBITDA and column: ' + column)
    plt.xlabel(column)
    plt.ylabel('EBITDA')

    # display the chart
    plt.show()
```

```
def getModel(my_dataset, column):
    # Clean-up: remove any outliers in the dataset
    print("Total rows before cleanup: {}".format(len(my_dataset)))
    z_scores_column = (my_dataset[column] - my_dataset[column].mean()) / my_dataset[column].std()
    z_scores_EBITDA = (my_dataset['EBITDA'] - my_dataset['EBITDA'].mean()) / my_dataset['EBITDA'].std()

    # define a std threshold
    threshold = 3

    outliers_mask = (np.abs(z_scores_column) > threshold) | (np.abs(z_scores_EBITDA) > threshold)
    my_dataset_no_outliers = my_dataset[~outliers_mask]

    # Clean-up: remove n/a values from columns
    my_dataset_no_outliers.dropna(inplace=True)

    print("Total rows after cleanup: {}".format(len(my_dataset_no_outliers)))

    # train the model using Linear Regression
    X = my_dataset_no_outliers.iloc[:, :-1].values
    y = my_dataset_no_outliers.iloc[:, -1].values
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)

    model = LinearRegression()
    model.fit(X_train, y_train)

    # test accuracy of Linear Regression model - model accuracy
    print("Analysis for column: " + column)
    score_logistic = model.score(X_test, y_test)
    print('Score of model:', score_logistic)

    # show coefficient and intercept
    coefficients = model.coef_
    intercept = model.intercept_

    print("Coefficients:", coefficients)
    print("Intercept:", intercept)

    y_pred = model.predict(X_test)

    mse = mean_squared_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)

    # print the results
    print('Mean squared error:', mse)
    print('R^2 score:', r2)

    return model
```

```
# X = Enterprise Value, Y = Ebitda
dataset1 = dataset.loc[:, ['Enterprise Value', 'EBITDA']]
visualizeBar(dataset1, 'Enterprise Value')
model1 = getModel(dataset1, 'Enterprise Value')
pickle.dump(model1, open('model1.model', 'wb'))
```



```
Total rows before cleanup: 1577
Total rows after cleanup: 1570
Analysis for column: Enterprise Value
Score of model: 0.392750358473443
Coefficients: [0.08914307]
Intercept: 1076582392.3549147
Mean squared error: 5.819498250556622e+22
R^2 score: 0.392750358473443
<ipython-input-8-8298406b9dfb>:14: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-vie
  my_dataset_no_outliers.dropna(inplace=True)
```

```
# Prediction of EBITDA based on hypothetical Enterprise Value

# Load the trained model
import pickle

model1 = pickle.load(open('model1.model', 'rb'))

# Define a new data point to make a prediction on
new_data = [[500000000]]

# Use the trained model to make a prediction on the new data point
predicted_ebitda = model1.predict(new_data)

print("Predicted EBITDA:", predicted_ebitda[0])
```

```
Predicted EBITDA: 1121153929.2554214
```

```
# Checking accuracy of EBITDA Vs. Enterprise Value using RMSE

# Load the necessary libraries and data
import numpy as np
import pandas as pd
import pickle
from sklearn.metrics import mean_squared_error

dataset = pd.read_excel('Final_ProjectData.xlsx', sheet_name='Sheet1', header=1)

# Load the trained model
model1 = pickle.load(open('model1.model', 'rb'))

# Extract the input features and the ground truth values from the dataset
X_test = dataset.loc[:, ['Enterprise Value']]
y_test = dataset.loc[:, ['EBITDA']]

# Use the trained model to make predictions on the test data
y_pred = model1.predict(X_test)

# Calculate the RMSE metric to evaluate the model's performance
rmse = np.sqrt(mean_squared_error(y_test, y_pred))

print("RMSE:", rmse)
```

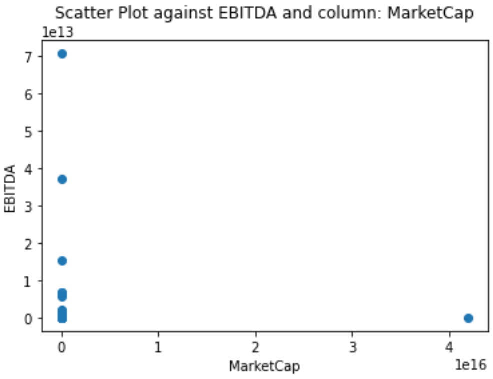
```
RMSE: 94308935286976.14
/usr/local/lib/python3.9/dist-packages/sklearn/base.py:432: UserWarning: X has feature names, but LinearRegression was fitted witho
  warnings.warn(
```

```
# Checking accuracy of EBITDA Vs. Enterprise Value using Logistic Regression Model

# Load the necessary libraries and data
import numpy as np
import pandas as pd
import pickle
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix

dataset = pd.read_excel('Final_ProjectData.xlsx', sheet_name='Sheet1', header=1)

# Define a binary target variable based on EBITDA values
dataset['EBITDA_binary'] = np.where(dataset['EBITDA'] > 0, 1, 0)

# Extract the input features and the binary target variable from the dataset
X = dataset.loc[:, ['Enterprise Value']]
y = dataset.loc[:, ['EBITDA_binary']]

# Split the data into training and testing sets
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train a logistic regression model on the training data
lr_model = LogisticRegression()
lr_model.fit(X_train, y_train)

# Use the trained logistic regression model to make predictions on the test data
y_pred = lr_model.predict(X_test)

# Evaluate the logistic regression model's accuracy on the test data
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)

print("Accuracy:", accuracy)
```

```
Accuracy: 0.16455696202531644
/usr/local/lib/python3.9/dist-packages/sklearn/utils/validation.py:1143: DataConversionWarning: A column-vector y was passed when a
  y = column_or_1d(y, warn=True)
/usr/local/lib/python3.9/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=
ABNORMAL_TERMINATION_IN_LNSRCH.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
# X = Market Cap, Y = Ebitda
dataset2 = dataset.loc[:, ['MarketCap', 'EBITDA']]
visualizeBar(dataset2, 'MarketCap')
```

```
model2 = getModel(dataset2, 'MarketCap')
pickle.dump(model2, open('model2.model', 'wb'))
```



```
Total rows before cleanup: 1577
Total rows after cleanup: 1570
Analysis for column: MarketCap
Score of model: -0.003236466033577523
Coefficients: [0.20538556]
Intercept: 2634869653.1740136
Mean squared error: 9.614386670200582e+22
R^2 score: -0.003236466033577523
<ipython-input-8-8298406b9dfb>:14: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-vie
  my_dataset_no_outliers.dropna(inplace=True)
```

```
# Prediction of EBITDA based on hypothetical MarketCap Value

# Load the trained model
import pickle

model1 = pickle.load(open('model1.model', 'rb'))

# Define a new data point to make a prediction on
new_data = [[500000000]]

# Use the trained model to make a prediction on the new data point
predicted_ebitda = model1.predict(new_data)

print("Predicted EBITDA:", predicted_ebitda[0])
```

```
Predicted EBITDA: 1121153929.2554214
```

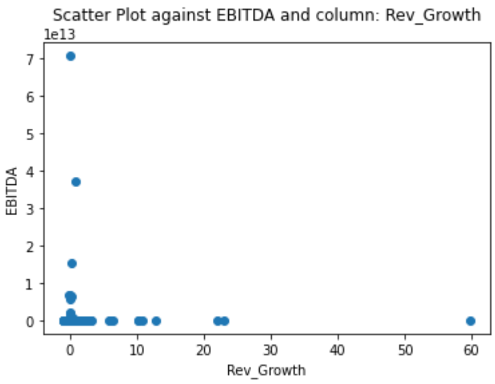
```
# Checking accuracy of EBITDA Vs. Market Cap using RMSE

# Load the necessary libraries and data
import numpy as np
import pandas as pd
import pickle
from sklearn.metrics import mean_squared_error

dataset = pd.read_excel('Final_ProjectData.xlsx', sheet_name='Sheet1', header=1)

# Load the trained model
model1 = pickle.load(open('model1.model', 'rb'))

# Extract the input features and the ground truth values from the dataset
X_test = dataset.loc[:, ['MarketCap']]
y_test = dataset.loc[:, ['EBITDA']]

# Use the trained model to make predictions on the test data
y_pred = model1.predict(X_test)

# Calculate the RMSE metric to evaluate the model's performance
rmse = np.sqrt(mean_squared_error(y_test, y_pred))

print("RMSE:", rmse)
```

```
RMSE: 94326586938646.86
/usr/local/lib/python3.9/dist-packages/sklearn/base.py:432: UserWarning: X has feature names, but LinearRegression was fitted witho
  warnings.warn(
```

```
# Checking accuracy of EBITDA Vs. MarketCap using Logistic Regression Model
# Load the necessary libraries and data
import numpy as np
import pandas as pd
import pickle
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix

dataset = pd.read_excel('Final_ProjectData.xlsx', sheet_name='Sheet1', header=1)

# Define a binary target variable based on EBITDA values
dataset['EBITDA_binary'] = np.where(dataset['EBITDA'] > 0, 1, 0)

# Extract the input features and the binary target variable from the dataset
X = dataset.loc[:, ['MarketCap']]
y = dataset.loc[:, ['EBITDA_binary']]

# Split the data into training and testing sets
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train a logistic regression model on the training data
lr_model = LogisticRegression()
lr_model.fit(X_train, y_train)

# Use the trained logistic regression model to make predictions on the test data
y_pred = lr_model.predict(X_test)

# Evaluate the logistic regression model's accuracy on the test data
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)

print("Accuracy:", accuracy)
```

```
Accuracy: 0.16455696202531644
/usr/local/lib/python3.9/dist-packages/sklearn/utils/validation.py:1143: DataConversionWarning: A column-vector y was passed when a
  y = column_or_1d(y, warn=True)
/usr/local/lib/python3.9/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=
ABNORMAL_TERMINATION_IN_LNSRCH.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
# X = Rev_Growth, Y = Ebitda
dataset3 = dataset.loc[:, ['Rev_Growth', 'EBITDA']]
visualizeBar(dataset3, 'Rev_Growth')
model3 = getModel(dataset3, 'Rev_Growth')
pickle.dump(model3, open('model3.model', 'wb'))
```



```
Total rows before cleanup: 1577
Total rows after cleanup: 1562
Analysis for column: Rev_Growth
Score of model: -0.0033247477197253517
Coefficients: [-1.73246279e+09]
Intercept: 6793668048.054706
Mean squared error: 9.610494763914514e+22
R^2 score: -0.0033247477197253517
<ipython-input-8-8298406b9dfb>:14: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-vie
  my_dataset_no_outliers.dropna(inplace=True)
```

```
# Prediction of EBITDA based on hypothetical Rev_Growth Value

# Load the trained model
import pickle

model1 = pickle.load(open('model1.model', 'rb'))

# Define a new data point to make a prediction on
new_data = [[0.555552887878415]]

# Use the trained model to make a prediction on the new data point
predicted_ebitda = model1.predict(new_data)

print("Predicted EBITDA:", predicted_ebitda[0])
```

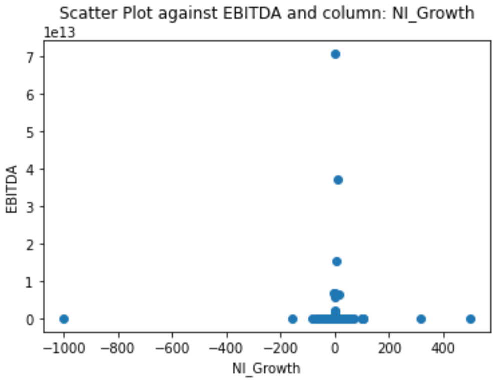
```
Predicted EBITDA: 1076582392.4044383
```

```
# Checking accuracy of EBITDA Vs. Rev-Growth using RMSE

# Load the necessary libraries and data
import numpy as np
import pandas as pd
import pickle
from sklearn.metrics import mean_squared_error

dataset = pd.read_excel('Final_ProjectData.xlsx', sheet_name='Sheet1', header=1)

# Load the trained model
model1 = pickle.load(open('model1.model', 'rb'))

# Extract the input features and the ground truth values from the dataset
X_test = dataset.loc[:, ['Rev_Growth']]
y_test = dataset.loc[:, ['EBITDA']]

# Use the trained model to make predictions on the test data
y_pred = model1.predict(X_test)

# Calculate the RMSE metric to evaluate the model's performance
rmse = np.sqrt(mean_squared_error(y_test, y_pred))

print("RMSE:", rmse)
```

```
RMSE: 2074091336610.3972
/usr/local/lib/python3.9/dist-packages/sklearn/base.py:432: UserWarning: X has feature names, but LinearRegression was fitted witho
  warnings.warn(
# Checking accuracy of EBITDA Vs. MarketCap using Logistic Regression Model

# Load the necessary libraries and data
import numpy as np
import pandas as pd
import pickle
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix

dataset = pd.read_excel('Final_ProjectData.xlsx', sheet_name='Sheet1', header=1)

# Define a binary target variable based on EBITDA values
dataset['EBITDA_binary'] = np.where(dataset['EBITDA'] > 0, 1, 0)

# Extract the input features and the binary target variable from the dataset
X = dataset.loc[:, ['Rev_Growth']]
y = dataset.loc[:, ['EBITDA_binary']]

# Split the data into training and testing sets
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train a logistic regression model on the training data
lr_model = LogisticRegression()
lr_model.fit(X_train, y_train)

# Use the trained logistic regression model to make predictions on the test data
y_pred = lr_model.predict(X_test)

# Evaluate the logistic regression model's accuracy on the test data
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)

print("Accuracy:", accuracy)
```

```
Accuracy: 0.8354430379746836
/usr/local/lib/python3.9/dist-packages/sklearn/utils/validation.py:1143: DataConversionWarning: A column-vector y was passed when a
  y = column_or_1d(y, warn=True)
```

```
# X = NI_Growth, Y = Ebitda
dataset4 = dataset.loc[:, ['NI_Growth', 'EBITDA']]
visualizeBar(dataset4, 'NI_Growth')
model4 = getModel(dataset4, 'NI_Growth')
pickle.dump(model4, open('model4.model', 'wb'))
```



```
Total rows before cleanup: 1577
Total rows after cleanup: 1564
Analysis for column: NI_Growth
Score of model: -0.003275792943949307
Coefficients: [95312588.15110844]
Intercept: 6308286912.054886
Mean squared error: 9.6080909250653e+22
R^2 score: -0.003275792943949307
<ipython-input-8-8298406b9dfb>:14: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-vie
  my_dataset_no_outliers.dropna(inplace=True)
```

```
# Prediction of EBITDA based on hypothetical NI_Growth Value

# Load the trained model
import pickle

model1 = pickle.load(open('model1.model', 'rb'))
```

```
# Define a new data point to make a prediction on
new_data = [[2.555552887878415]]

# Use the trained model to make a prediction on the new data point
predicted_ebitda = model1.predict(new_data)

print("Predicted EBITDA:", predicted_ebitda[0])
```

```
    Predicted EBITDA: 1076582392.5827246
```

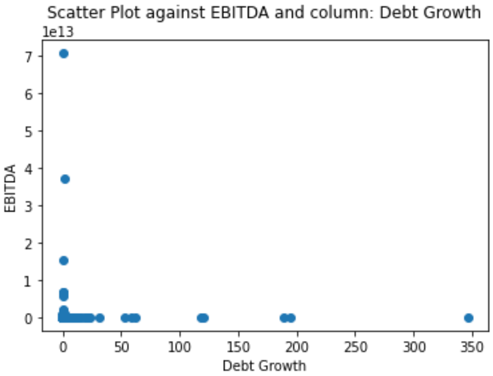
```
# Checking accuracy of EBITDA Vs. NI_Growth using RMSE

# Load the necessary libraries and data
import numpy as np
import pandas as pd
import pickle
from sklearn.metrics import mean_squared_error

dataset = pd.read_excel('Final_ProjectData.xlsx', sheet_name='Sheet1', header=1)

# Load the trained model
model1 = pickle.load(open('model1.model', 'rb'))

# Extract the input features and the ground truth values from the dataset
X_test = dataset.loc[:, ['NI_Growth']]
y_test = dataset.loc[:, ['EBITDA']]

# Use the trained model to make predictions on the test data
y_pred = model1.predict(X_test)

# Calculate the RMSE metric to evaluate the model's performance
rmse = np.sqrt(mean_squared_error(y_test, y_pred))

print("RMSE:", rmse)
```

```
    RMSE: 2074091336610.384
    /usr/local/lib/python3.9/dist-packages/sklearn/base.py:432: UserWarning: X has feature names, but LinearRegression was fitted witho
      warnings.warn(
```

```
# Checking accuracy of EBITDA Vs. NI_Growth using Logistic Regression Model

# Load the necessary libraries and data
import numpy as np
import pandas as pd
import pickle
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix

dataset = pd.read_excel('Final_ProjectData.xlsx', sheet_name='Sheet1', header=1)

# Define a binary target variable based on EBITDA values
dataset['EBITDA_binary'] = np.where(dataset['EBITDA'] > 0, 1, 0)

# Extract the input features and the binary target variable from the dataset
X = dataset.loc[:, ['NI_Growth']]
y = dataset.loc[:, ['EBITDA_binary']]

# Split the data into training and testing sets
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train a logistic regression model on the training data
lr_model = LogisticRegression()
lr_model.fit(X_train, y_train)

# Use the trained logistic regression model to make predictions on the test data
y_pred = lr_model.predict(X_test)

# Evaluate the logistic regression model's accuracy on the test data
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)

print("Accuracy:", accuracy)
```

```
    Accuracy: 0.8354430379746836
    /usr/local/lib/python3.9/dist-packages/sklearn/utils/validation.py:1143: DataConversionWarning: A column-vector y was passed when a
      y = column_or_1d(y, warn=True)
```

```
# X = Debt Growth, Y = Ebitda
dataset5 = dataset.loc[:, ['Debt Growth', 'EBITDA']]
visualizeBar(dataset5, 'Debt Growth')
model5 = getModel(dataset5, 'Debt Growth')
pickle.dump(model5, open('model5.model', 'wb'))
```



```
Total rows before cleanup: 1577
Total rows after cleanup: 1563
Analysis for column: Debt Growth
Score of model: -0.002975689680487781
Coefficients: [-2.16975912e+08]
Intercept: 6735243303.39075
Mean squared error: 9.593335057864696e+22
R^2 score: -0.002975689680487781
<ipython-input-8-8298406b9dfb>:14: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-vie
  my_dataset_no_outliers.dropna(inplace=True)
```

```
# Prediction of EBITDA based on hypothetical Debt Growth Value

# Load the trained model
import pickle

model1 = pickle.load(open('model1.model', 'rb'))

# Define a new data point to make a prediction on
new_data = [[-1.555552887878415]]

# Use the trained model to make a prediction on the new data point
predicted_ebitda = model1.predict(new_data)

print("Predicted EBITDA:", predicted_ebitda[0])
```

```
Predicted EBITDA: 1076582392.2162478
```

```
# Checking accuracy of EBITDA Vs. Debt Growth using RMSE

# Load the necessary libraries and data
import numpy as np
import pandas as pd
import pickle
from sklearn.metrics import mean_squared_error

dataset = pd.read_excel('Final_ProjectData.xlsx', sheet_name='Sheet1', header=1)

# Load the trained model
model1 = pickle.load(open('model1.model', 'rb'))

# Extract the input features and the ground truth values from the dataset
X_test = dataset.loc[:, ['Debt Growth']]
y_test = dataset.loc[:, ['EBITDA']]

# Use the trained model to make predictions on the test data
y_pred = model1.predict(X_test)

# Calculate the RMSE metric to evaluate the model's performance
rmse = np.sqrt(mean_squared_error(y_test, y_pred))

print("RMSE:", rmse)
```

```
RMSE: 2074091336610.3972
/usr/local/lib/python3.9/dist-packages/sklearn/base.py:432: UserWarning: X has feature names, but LinearRegression was fitted witho
  warnings.warn(
# Checking accuracy of EBITDA Vs. Debt Growth using Logistic Regression Model

# Load the necessary libraries and data
import numpy as np
import pandas as pd
import pickle
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix

dataset = pd.read_excel('Final_ProjectData.xlsx', sheet_name='Sheet1', header=1)

# Define a binary target variable based on EBITDA values
dataset['EBITDA_binary'] = np.where(dataset['EBITDA'] > 0, 1, 0)

# Extract the input features and the binary target variable from the dataset
X = dataset.loc[:, ['Debt Growth']]
y = dataset.loc[:, ['EBITDA_binary']]

# Split the data into training and testing sets
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train a logistic regression model on the training data
lr_model = LogisticRegression()
lr_model.fit(X_train, y_train)

# Use the trained logistic regression model to make predictions on the test data
y_pred = lr_model.predict(X_test)

# Evaluate the logistic regression model's accuracy on the test data
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)

print("Accuracy:", accuracy)
```

```
Accuracy: 0.8322784810126582
/usr/local/lib/python3.9/dist-packages/sklearn/utils/validation.py:1143: DataConversionWarning: A column-vector y was passed when a
  y = column_or_1d(y, warn=True)
```

```
# Apply SMOTE to balance dataset (using categorical data)
dataset_smote = dataset.loc[:, ['EBITDA', 'Rating']]

# train the model using Linear Regression
X = dataset_smote.iloc[:, :-1].values
y = dataset_smote.iloc[:, -1].values

smote = SMOTE()
X_resampled, y_resampled = smote.fit_resample(X, Y)

X_train, X_test, y_train, y_test = train_test_split(X_resampled, y_resampled, test_size = 0.2, random_state = 0)

# train the model using Linear Regression
model_smote = LogisticRegression()
model_smote.fit(X_train, y_train)

# test accuracy of Logistic Regression model - model accuracy
score_logistic = model_smote.score(X_test, y_test)
print(score_logistic)

pickle.dump(model_smote, open('smote_model.model', 'wb'))
```

```
0.43523316062176165
```

```
# fraud or not fraud
fraud_notfraud_df = dataset.loc[:, ['MScore', 'Fraud or Non Fraud']]
X = fraud_notfraud_df.iloc[:, :-1].values
Y = fraud_notfraud_df.iloc[:, -1].values

X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 0.2, random_state = 0)

# train the model using Linear Regression
```