Keara Jones
A00325654

1. Regression

1.1 Objectives (Business Understanding)

The abalone dataset from the UCI machine learning repository was used for the regression analysis. The goal of the model is to predict the number of rings, which is the age of the abalone, using features including length, diameter, and weights.After carrying out a linear regression analysis, polynomial terms were added to extend the model.

1.2 Data Exploration (Data Understanding & Preparation)

The abalone dataset has the features Sex, Length, Diameter, Weight, Whole weight, Shucked weight, Viscera weight, and Shell weight. The target variable is Rings, which represents the age of the abalone.

	Sex	Length	Diameter	Height	Whole_weight	Shucked_weight	Viscera_weight	Shell_weight	Rings
0	M	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.150	15
1	M	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.070	7
2	F	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.210	9
3	M	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.155	10
4	I	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.055	7

Sex was categorical with values M, F, and I (Infant). Using one hot encoding, to separate into columns Sex_M and Sex_I, and dropping Sex_F to prevent multicollinearity.

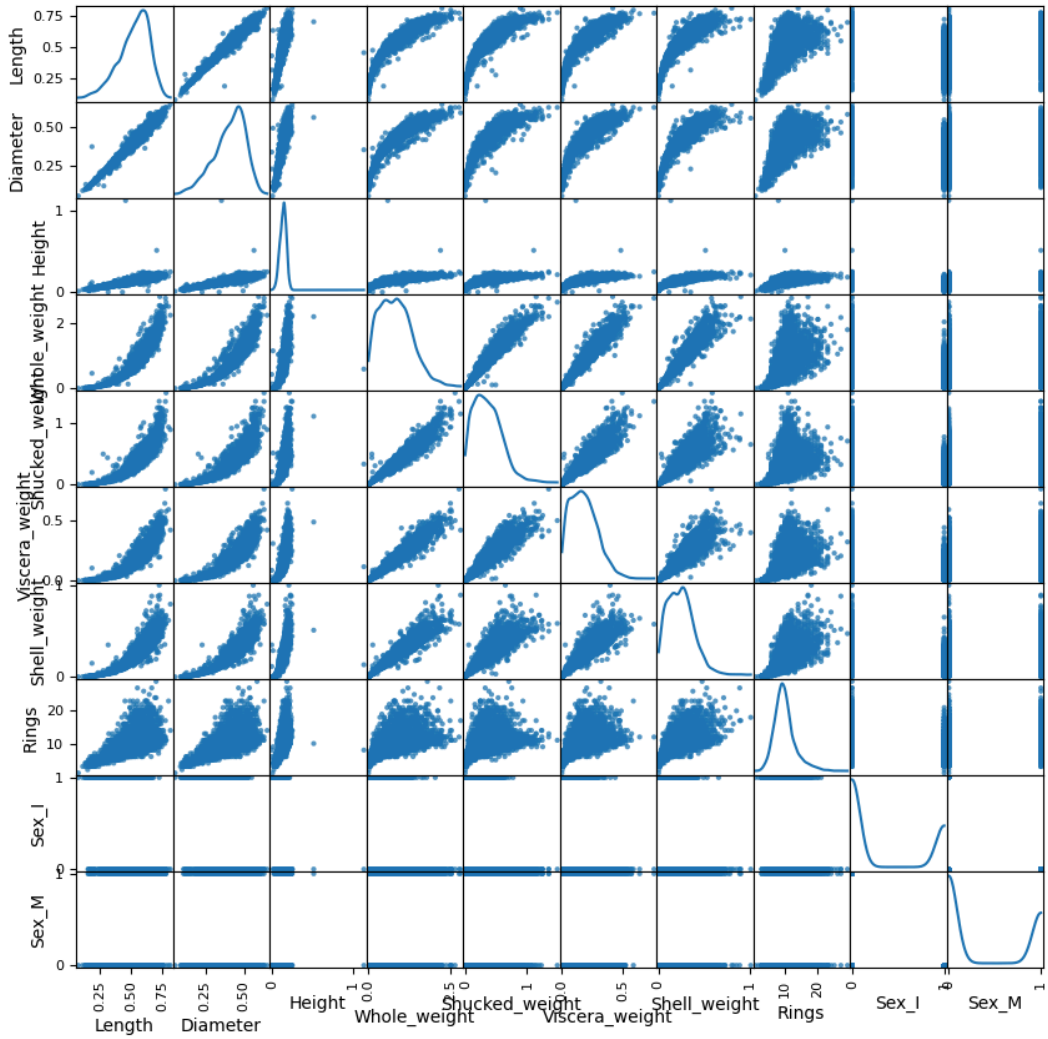
```
df = pd.get_dummies(df, columns=['Sex'], drop_first=True)
df.head()
```

	Length	Diameter	Height	Whole_weight	Shucked_weight	Viscera_weight	Shell_weight	Rings	Sex_I	Sex_M
0	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.150	15	False	True
1	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.070	7	False	True
2	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.210	9	False	False
3	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.155	10	False	True
4	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.055	7	True	False

The dataset showed significant variability among features, particularly between the weight measurements and height, emphasizing the need for feature scaling. Correlation analysis revealed moderate positive relationships between rings and whole_weight and shell_weight, suggesting these as strong predictors, while height had a weaker correlation, indicating that it is not a major predictor.

	Length	Diameter	Height	Whole_weight	Shucked_weight	Viscera_weight	Shell_weight	Rings	Sex_I	Sex_M
Length	1.000000	0.986812	0.827554	0.925261	0.897914	0.903018	0.897706	0.556720	-0.551465	0.236543
Diameter	0.986812	1.000000	0.833684	0.925452	0.893162	0.899724	0.905330	0.574660	-0.564315	0.240376
Height	0.827554	0.833684	1.000000	0.819221	0.774972	0.798319	0.817338	0.557467	-0.518552	0.215459
Whole_weight	0.925261	0.925452	0.819221	1.000000	0.969405	0.966375	0.955355	0.540390	-0.557592	0.252038
Shucked_weight	0.897914	0.893162	0.774972	0.969405	1.000000	0.931961	0.882617	0.420884	-0.521842	0.251793
Viscera_weight	0.903018	0.899724	0.798319	0.966375	0.931961	1.000000	0.907656	0.503819	-0.556081	0.242194
Shell_weight	0.897706	0.905330	0.817338	0.955355	0.882617	0.907656	1.000000	0.627574	-0.546953	0.235391
Rings	0.556720	0.574660	0.557467	0.540390	0.420884	0.503819	0.627574	1.000000	-0.436063	0.181831
Sex_I	-0.551465	-0.564315	-0.518552	-0.557592	-0.521842	-0.556081	-0.546953	-0.436063	1.000000	-0.522541
Sex_M	0.236543	0.240376	0.215459	0.252038	0.251793	0.242194	0.235391	0.181831	-0.522541	1.000000

The scatter plot shows positive trends between rings and whole weight, shucked weight, viscera weight, and shell weight. Height showed weak and inconsistent relationships with rings, suggesting it has little predictive value. The varying scatter patterns showed non-linear relationships suggesting the need for polynomial transformations.



1.3 Modeling

A linear regression model was built

```
model = LinearRegression()
model.fit(X_train, y_train)
print('intercept:', model.intercept_)
print('slope:', model.coef_)
print('R^2:', model.score(X_train, y_train))
```

```
intercept: 3.91085614463705
slope: [ -1.45240996  12.65718031   9.16105749   8.48824635 -19.41954003
  -9.07097611   8.83729616  -0.8095619    0.09705978]
R^2: 0.5376900347266163
```

```
yhat = model.predict(X_test)
print('RMSE', mean_squared_error(y_test, yhat, squared=False))
```

```
RMSE 2.1831074597330833
```

The R^2 was 0.5377 and the RMSE was 2.1831.

```
X['SexMsquared'] = np.square(df.Sex_M)
X['SexIsquared'] = np.square(df.Sex_I)
X['Lenghtsquared'] = np.square(df.Length)
X['Diametersquared'] = np.square(df.Diameter)
X['Heightsquared'] = np.square(df.Height)
X['Whole_weightsqaured'] = np.square(df.Whole_weight)
X['Viscera_weightsqaured'] = np.square(df.Viscera_weight)
X['Shell_weightsqaured'] = np.square(df.Shell_weight)
X.head()
```

After squaring the terms the R^2 was 0.5577 and the RMSE was 2.1335.

1.4 Evaluation

After applying polynomial regression the R^2 increased and the RMSE decreased which shows a slight improvement in the model. There is still room for improvement and the model could be improved by using higher-degree polynomials or other techniques.

2. Decision Tree

2.1 Objectives (Business Understanding)

For the decision tree the Maternal Health Risk dataset, from the UCI machine learning repository, was used. The dataset contains maternal health data from rural areas in Bangladesh. The goal of the model is to predict the risk level of maternal mortality based on these features.

2.2 Data Exploration (Data Understanding & Preparation)

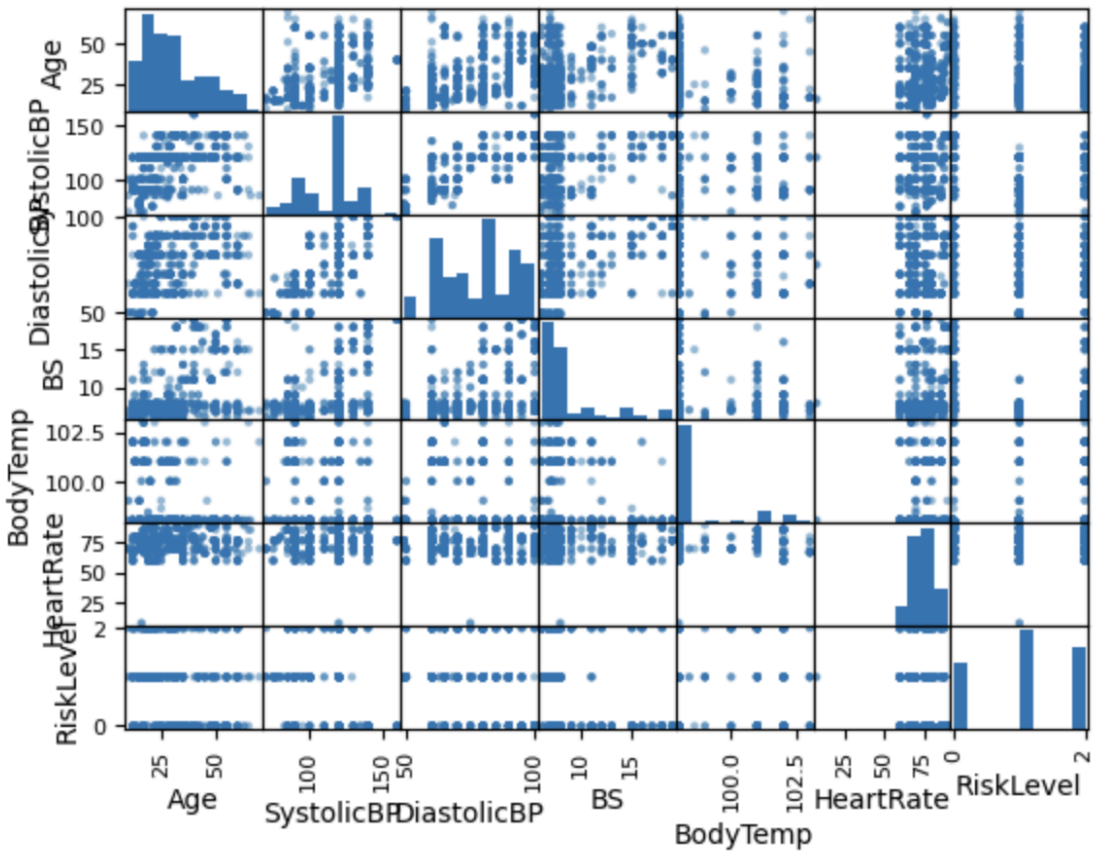
The dataset has features Age, SystolicBP, Diastolic BP, Blood Sugar, Body Temp, Heart Rate, and Risk Level. Risk level as the target variable with values low risk, mid risk, and high risk

	Age	SystolicBP	DiastolicBP	BS	BodyTemp	HeartRate	RiskLevel
0	25	130	80	15.0	98.0	86	high risk
1	35	140	90	13.0	98.0	70	high risk
2	29	90	70	8.0	100.0	80	high risk
3	30	140	85	7.0	98.0	70	high risk
4	35	120	60	6.1	98.0	76	low risk

The most common category is low risk. Mid-risk and high-risk still make up a large percentage.

```
RiskLevel
low risk     406
mid risk     336
high risk    272
Name: count, dtype: int64
```

The scatter plot shows that blood pressure shows a positive relationship with risk, with high-risk individuals often having higher blood pressure. Features like blood pressure, blood sugar, and heart rate appear to be indicators of maternal health risk. Other features do not show a relationship with risk.



2.3 Modeling

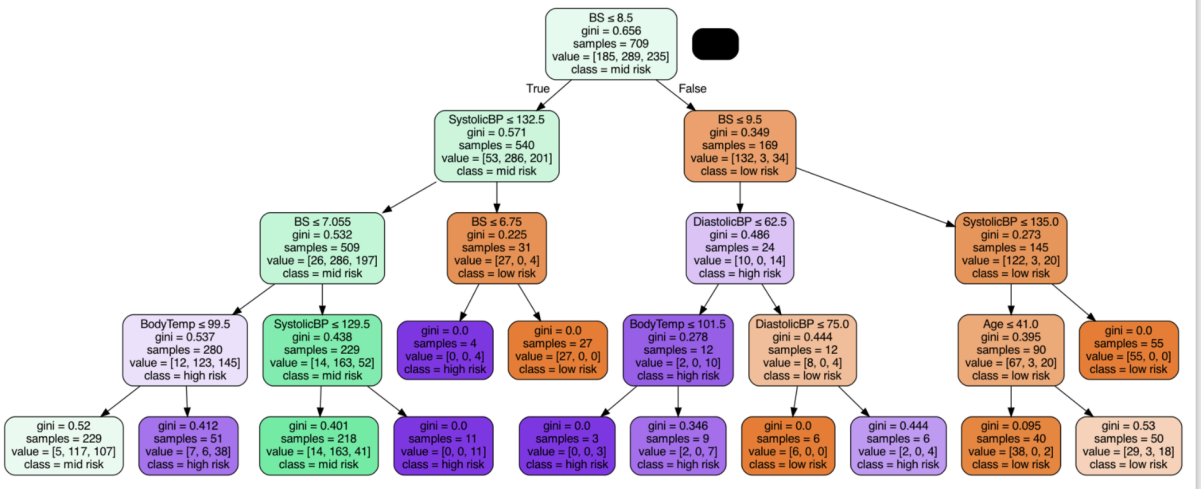
The data was split into training and test data:
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state=1)
Then to find the best depth for the decision tree cross validation was used.

```
|: for d in range(1,20) :
       model = DecisionTreeClassifier(max_depth = d )
       scores = cross_val_score(model, X_train, y_train, cv=5)
       print("d: ", d, " val accuracy: ", scores.mean())
```

```
d:  1  val accuracy:  0.5810708220956947
d:  2  val accuracy:  0.6191689141943861
d:  3  val accuracy:  0.6601038857257018
d:  4  val accuracy:  0.6953451203675955
d:  5  val accuracy:  0.7080811107781442
d:  6  val accuracy:  0.7150634302267507
d:  7  val accuracy:  0.7432624113475177
d:  8  val accuracy:  0.7503645989411647
d:  9  val accuracy:  0.7687144141444411
d:  10  val accuracy:  0.7587753471181701
d:  11  val accuracy:  0.7786135251223654
d:  12  val accuracy:  0.8025471980821097
d:  13  val accuracy:  0.8166716611727101
d:  14  val accuracy:  0.8138347817400859
d:  15  val accuracy:  0.818060133852762
d:  16  val accuracy:  0.8067825392068724
d:  17  val accuracy:  0.8124263310358606
d:  18  val accuracy:  0.8152432324443112
d:  19  val accuracy:  0.8081810008990111
```

The decision tree model was built with a maximum depth of 15. A decision tree with a maximum depth of 4 is shown for the report.

120]:



2.4 Evaluation

A training accuracy of 92% and test accuracy of 80% show that the model performs well on the training data but struggles with new data, which may be because of overfitting. The drop in the performance on the test data shows the model struggles to generalize unseen data.

```
print("Training Accuracy:", model.score(X_train, y_train))
print("Test Accuracy:", model.score(X_test, y_test))
```

```
Training Accuracy: 0.9294781382228491
Test Accuracy: 0.8065573770491803
```

```
y_hat = model.predict(X_test)
print("Test Accuracy:", accuracy_score(y_test, y_hat))
```

```
Test Accuracy: 0.8065573770491803
```

```
cm = confusion_matrix(y_test, y_hat)
print(cm)
```

```
[[79  5  3]
 [ 4 95 18]
 [12 17 72]]
```

3 kNN

3.3 Modeling

The data was split into training and test data. A kNN classifier was first built with the unscaled data.

```
clf = KNeighborsClassifier()
scores = cross_val_score(clf, X_train, y_train, cv=5)

print('Validation Accuracy, unscaled', scores.mean())
```

The validation accuracy for the unscaled data was 0.7024572969733294

```
scaler = MinMaxScaler()
scaler.fit(X_train)
X_train_scaled = scaler.transform(X_train)
X_test_scaled = scaler.transform(X_test)

scores = cross_val_score(clf, X_train_scaled, y_train, cv=5)

print('Validation Accuracy, min-max', scores.mean())
```

```
Validation Accuracy, min-max 0.7024572969733294
```

Then min-max scaling was applied and the validation accuracy was 0.7024572969733294. After min-max scaling was applied the validation accuracy was the same.

```
scaler = StandardScaler()
scaler.fit(X_train)
X_train_scaled = scaler.transform(X_train)

scores = cross_val_score(clf, X_train_scaled, y_train, cv=5)

print('Validation Accuracy, z-scaled', scores.mean())
```

```
Validation Accuracy, z-scaled 0.6784437119168915
```

Z-scaling was applied and the validation accuracy, 0.6784437119168915 was lower than unscaled and min-max scaled. Scaling does not improve the performance.

Cross-validation was used to find the best k, the best k value being 1.

```
for k in range(1,15) :
    clf = KNeighborsClassifier(n_neighbors=k)
    scores = cross_val_score(clf, X_train, y_train, cv=5)
    print("k: ", k, "validation accuracy: ", scores.mean())
```

```
k:  1 validation accuracy:  0.7954749775247228
k:  2 validation accuracy:  0.7404754769753271
k:  3 validation accuracy:  0.70524423134552
k:  4 validation accuracy:  0.7010188792328439
k:  5 validation accuracy:  0.7024572969733294
k:  6 validation accuracy:  0.7052642093696934
k:  7 validation accuracy:  0.6996204175407051
k:  8 validation accuracy:  0.6911597243032664
k:  9 validation accuracy:  0.6813105583857757
k:  10 validation accuracy:  0.6855159324742783
k:  11 validation accuracy:  0.6770452502247528
k:  12 validation accuracy:  0.678423733892718
k:  13 validation accuracy:  0.6713814803715913
k:  14 validation accuracy:  0.6713814803715913
```

A kNN classifier was built with 1 neighbor.

```
clf = KNeighborsClassifier(n_neighbors = 1)
clf.fit(X_train,y_train)
print('Test Accuracy: ', clf.score(X_test, y_test))
```

```
Test Accuracy:  0.7934426229508197
```

3.4 Evaluation

Scaling the data showed no improvements in performance. The validation accuracy stayed the same with min-max scaling and decreased with z-scaling. Since the best value for k was 1, it shows the model is too sensitive and does not work well with the unseen data. There may be overfitting to the training data. A test accuracy of 0.7934426229508197 shows there could be an improvement in the model.