

Abstract

The goal of this project is to explore the capabilities of various machine learning algorithms for predicting penguin species in the Palmer penguins data set. To do so, the work was split into three main sections: an exploration, modelling, and an evaluation. The exploration section is meant to develop an understanding of the dataset through various visualizations as well as summary tables of the dataset. The idea is that a better understanding of the dataset would lead to better feature selection for the modelling section. Specifically, SciKitLearn's feature selection tools were employed to find the features that were the most helpful. Once the features were selected, several models were built, including a Logistic Regression, Decision Tree, Support Vector Machine, and Random Forest. 5-Fold cross validation was used to both evaluate the performance of the models and find the best parameters. Finally, the models were evaluated on the test set, with a Random Forest acheiving an accuracy of 100%.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.patches import Patch
from sklearn.preprocessing import LabelEncoder
import seaborn as sns
from sklearn.feature_selection import SelectKBest, chi2, f_classif, mutual_info_classif
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.model_selection import cross_val_score
from sklearn.metrics import confusion_matrix

train_url = "https://raw.githubusercontent.com/PhilChodrow/ml-notes/main/data/palmer-penguins"
df_train = pd.read_csv(train_url)

le = LabelEncoder()
le.fit(df_train["Species"])

def prepare_data(df):
    df = df.drop(["StudyName", "Sample Number", "Individual ID", "Date Egg", "Comments", "
    df = df[df["Sex"] != "."]
    df = df.dropna()
    y = le.transform(df["Species"])
    df = df.drop(["Species"], axis = 1)
    df = pd.get_dummies(df)
    return df, y

X_train, y_train = prepare_data(df_train)
X_train.head()
```

	Culmen Length (mm)	Culmen Depth (mm)	Flipper Length (mm)	Body Mass (g)	Delta 15 N (o/oo)	Delta 13 C (o/oo)	Island_Biscoe	Island_Dream	Island_Torgersen	Stage_Adult	1 Egg Stage	Continent
0	40.9	16.6	187.0	3200.0	9.08458	-24.54903	False	True	False	True	False	Antarctica
1	49.0	19.5	210.0	3950.0	9.53262	-24.66867	False	True	False	True	False	Antarctica
2	50.0	15.2	218.0	5700.0	8.25540	-25.40075	True	False	False	True	False	Antarctica
3	45.8	14.6	210.0	4200.0	7.79958	-25.62618	True	False	False	True	False	Antarctica
4	51.0	18.8	203.0	4100.0	9.23196	-24.17282	False	True	False	True	False	Antarctica

Feature Selection

```
# FIND THE BEST OVERALL 3 FEATURES
X_train_k_best_features_m = SelectKBest(mutual_info_classif, k=4).fit(X_train, y_train)
X_train_k_best_features_m.get_feature_names_out()

array(['Culmen Length (mm)', 'Culmen Depth (mm)', 'Flipper Length (mm)',
       'Delta 13 C (o/oo)'], dtype=object)

# FIND THE BEST 2 NUMERICAL FEATURES
numerical_cols = ['Culmen Length (mm)', 'Culmen Depth (mm)', 'Flipper Length (mm)', 'Body Mass (g)']
X_train_k_best_features_f = SelectKBest(f_classif, k=2).fit(X_train[numerical_cols], y_train)
X_train_k_best_features_f.get_feature_names_out()

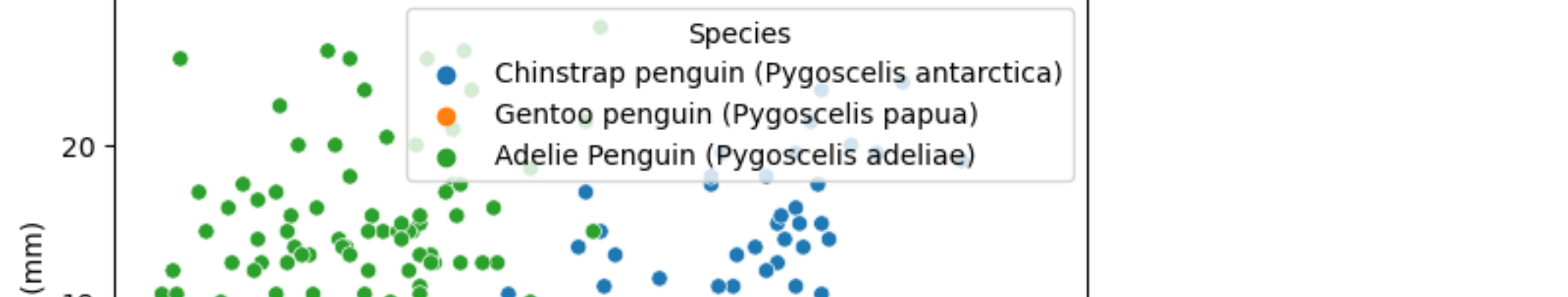
array(['Culmen Length (mm)', 'Flipper Length (mm)'], dtype=object)

# FIND THE BEST CATEGORICAL FEATURE
categorical_cols = ['Island_Biscoe', 'Island_Dream', 'Island_Torgersen', 'Stage_Adult', '1 Egg Stage']
X_train_k_best_features_c = SelectKBest(chi2, k=3).fit(X_train[categorical_cols], y_train)
X_train_k_best_features_c.get_feature_names_out()

array(['Island_Biscoe', 'Island_Dream', 'Island_Torgersen'], dtype=object)
```

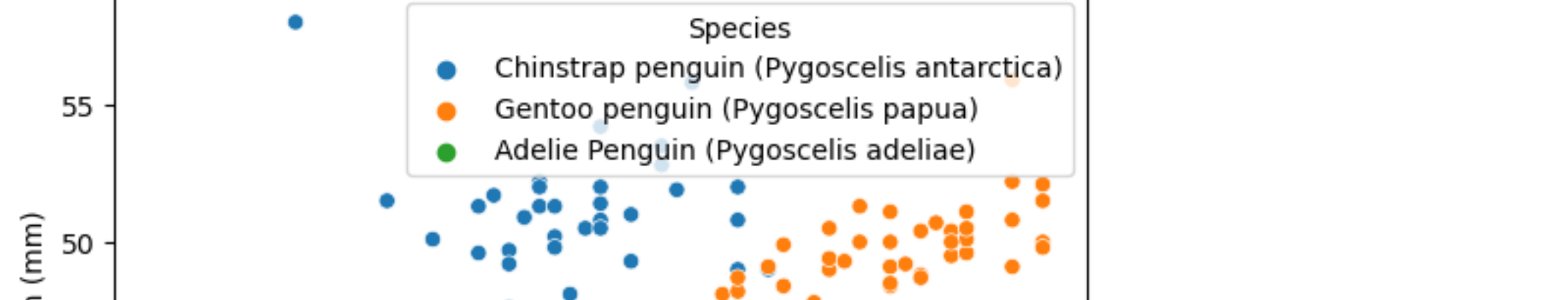
My first task was to find the 3 best features to use for classification, which I opted to do with scikit-learns SelectKBest function. This takes in a scoring function, as well as the training features and target variable. I first ran this with k=3 and mutual information scoring to see which 3 features, among all numerical and categorical features, have the lowest dependency. This is desirable because, since we can only use 3 features, we want each feature to give us new information about the target variable. If the three features are highly dependent, then they are redundant in their information. Using mutual information scoring, I saw that the 3 features with the lowest mutual dependency were 'Culmen Length (mm)', 'Culmen Depth (mm)', and 'Flipper Length (mm)'. These 3 features were all numerical, and since I knew I needed a combination of 2 numerical and 1 categorical feature, I split my feature selection between the numerical features and the categorical features. When scoring the numerical features, I opted to use analysis of variance F-value to find the degree of linear dependency. I again saw that 'Culmen Length (mm)' and 'Flipper Length (mm)' were the least linearly dependent features. When scoring the categorical features, I used chi-squared to measure the dependency between each feature and the target variable so see which was most relevant for classification. I found that the 'Island' feature, which had been split into 'Island_Biscoe', 'Island_Dream', and 'Island_Torgersen' during data processing were the most informative.

Visualizations



After selecting my variables I went to visualize some of them. I first started with the relationship between 'Culmen Length (mm)' and 'Flipper Length (mm)' and species. From looking at the graph I saw that the three species have pretty unique relationships between the culmen length and depth.

'Unnamed: 0' is no longer in the dataset, but when it was... I think it represented each penguin's ID from the data collection which got converted into a feature. Moreover, the ID's seemed to be in order of species; in other words, Adelle Penguins are ID's 0 - 150ish, Gentoo's are 150ish - 215ish, and Chinstraps are 215ish - 350. This meant that the data are perfectly linearly separable by ID alone, and therefore Support Vector Machines, Decision Trees, and even a Perceptron should be able to perfectly classify the training data by ID alone, and possibly the testing data as well if the testing data ID's match this distribution.



I then wanted to visualize the relationship between 'Flipper Length (mm)' and 'Culmen Length (mm)' and species. This plot shows that again there is a pretty unique relationship for each species between the two measurements.

```
frequency_tab = pd.crosstab(df_train['Species'], df_train['Island'])
frequency_tab
```

Island	Biscoe	Dream	Torgersen
Species			
Adelle Penguin (Pygoscelis adeliae)	33	45	42
Chinstrap penguin (Pygoscelis antarctica)	0	57	0
Gentoo penguin (Pygoscelis papua)	98	0	0

Finally, to get an understanding of the relationship between the Island and the species, I decided to make a frequency table showing the counts of each specie on each Island. The Pandas crosstab function makes this really easy to do. I saw that while Adelle penguins are found on all three islands, Chinstraps are found only on the Dream Island and Gentoo's are found only on the Biscoe Island.

Building the Models

```
predictor_cols = ['Culmen Length (mm)', 'Culmen Depth (mm)', 'Island_Dream', 'Island_Biscoe']

LR = LogisticRegression(max_iter=10000)
LR.fit(X_train[predictor_cols], y_train)
LR.score(X_train[predictor_cols], y_train)

0.99609375
```

```
scores = []
gammas = 10**np.arange(float(-5), float(5))

for gamma in gammas:
    SVM = SVC(gamma=gamma)
    SVM.fit(X_train[predictor_cols], y_train)

    cv_scores_SVM = cross_val_score(SVM, X_train[predictor_cols], y_train, cv=5)
    scores.append((gamma, cv_scores_SVM.mean()))

print(scores)

SVM = SVC(gamma=0.001)
SVM.fit(X_train[predictor_cols], y_train)
SVM.score(X_train[predictor_cols], y_train)

[(1e-05, 0.42187028657616893), (0.0001, 0.7266214177978884), (0.001, 0.9217948717948717), (0.01, 0.94140625), (0.1, 0.95703125), (1, 0.9647812), (10, 0.967028657616894), (100, 0.9678125), (1000, 0.9678125), (10000, 0.9678125)]
```

```
DT = DecisionTreeClassifier(max_depth=2)
DT.fit(X_train[predictor_cols], y_train)
DT.score(X_train[predictor_cols], y_train)

[(1, 0.7147058823529412), (2, 0.9294871794871794), (3, 0.9687028657616894), (4, 0.9647812), (5, 0.9647812), (6, 0.9647812), (7, 0.9647812), (8, 0.9647812), (9, 0.9647812), (10, 0.9647812), (11, 0.9647812), (12, 0.9647812), (13, 0.9647812), (14, 0.9647812), (15, 0.9647812), (16, 0.9647812), (17, 0.9647812), (18, 0.9647812), (19, 0.9647812), (20, 0.9647812), (21, 0.9647812), (22, 0.9647812), (23, 0.9647812), (24, 0.9647812), (25, 0.9647812), (26, 0.9647812), (27, 0.9647812), (28, 0.9647812), (29, 0.9647812), (30, 0.9647812), (31, 0.9647812), (32, 0.9647812), (33, 0.9647812), (34, 0.9647812), (35, 0.9647812), (36, 0.9647812), (37, 0.9647812), (38, 0.9647812), (39, 0.9647812), (40, 0.9647812), (41, 0.9647812), (42, 0.9647812), (43, 0.9647812), (44, 0.9647812), (45, 0.9647812), (46, 0.9647812), (47, 0.9647812), (48, 0.9647812), (49, 0.9647812), (50, 0.9647812), (51, 0.9647812), (52, 0.9647812), (53, 0.9647812), (54, 0.9647812), (55, 0.9647812), (56, 0.9647812), (57, 0.9647812), (58, 0.9647812), (59, 0.9647812), (60, 0.9647812), (61, 0.9647812), (62, 0.9647812), (63, 0.9647812), (64, 0.9647812), (65, 0.9647812), (66, 0.9647812), (67, 0.9647812), (68, 0.9647812), (69, 0.9647812), (70, 0.9647812), (71, 0.9647812), (72, 0.9647812), (73, 0.9647812), (74, 0.9647812), (75, 0.9647812), (76, 0.9647812), (77, 0.9647812), (78, 0.9647812), (79, 0.9647812), (80, 0.9647812), (81, 0.9647812), (82, 0.9647812), (83, 0.9647812), (84, 0.9647812), (85, 0.9647812), (86, 0.9647812), (87, 0.9647812), (88, 0.9647812), (89, 0.9647812), (90, 0.9647812), (91, 0.9647812), (92, 0.9647812), (93, 0.9647812), (94, 0.9647812), (95, 0.9647812), (96, 0.9647812), (97, 0.9647812), (98, 0.9647812), (99, 0.9647812), (100, 0.9647812), (101, 0.9647812), (102, 0.9647812), (103, 0.9647812), (104, 0.9647812), (105, 0.9647812), (106, 0.9647812), (107, 0.9647812), (108, 0.9647812), (109, 0.9647812), (110, 0.9647812), (111, 0.9647812), (112, 0.9647812), (113, 0.9647812), (114, 0.9647812), (115, 0.9647812), (116, 0.9647812), (117, 0.9647812), (118, 0.9647812), (119, 0.9647812), (120, 0.9647812), (121, 0.9647812), (122, 0.9647812), (123, 0.9647812), (124, 0.9647812), (125, 0.9647812), (126, 0.9647812), (127, 0.9647812), (128, 0.9647812), (129, 0.9647812), (130, 0.9647812), (131, 0.9647812), (132, 0.9647812), (133, 0.9647812), (134, 0.9647812), (135, 0.9647812), (136, 0.9647812), (137, 0.9647812), (138, 0.9647812), (139, 0.9647812), (140, 0.9647812), (141, 0.9647812), (142, 0.9647812), (143, 0.9647812), (144, 0.9647812), (145, 0.9647812), (146, 0.9647812), (147, 0.9647812), (148, 0.9647812), (149, 0.9647812), (150, 0.9647812), (151, 0.9647812), (152, 0.9647812), (153, 0.9647812), (154, 0.9647812), (155, 0.9647812), (156, 0.9647812), (157, 0.9647812), (158, 0.9647812), (159, 0.9647812), (160, 0.9647812), (161, 0.9647812), (162, 0.9647812), (163, 0.9647812), (164, 0.9647812), (165, 0.9647812), (166, 0.9647812), (167, 0.9647812), (168, 0.9647812), (169, 0.9647812), (170, 0.9647812), (171, 0.9647812), (172, 0.9647812), (173, 0.9647812), (174, 0.9647812), (175, 0.9647812), (176, 0.9647812), (177, 0.9647812), (178, 0.9647812), (179, 0.9647812), (180, 0.9647812), (181, 0.9647812), (182, 0.9647812), (183, 0.9647812), (184, 0.9647812), (185, 0.9647812), (186, 0.9647812), (187, 0.9647812), (188, 0.9647812), (189, 0.9647812), (190, 0.9647812), (191, 0.9647812), (192, 0.9647812), (193, 0.9647812), (194, 0.9647812), (195, 0.9647812), (196, 0.9647812), (197, 0.9647812), (198, 0.9647812), (199, 0.9647812), (200, 0.9647812), (201, 0.9647812), (202, 0.9647812), (203, 0.9647812), (204, 0.9647812), (205, 0.9647812), (206, 0.9647812), (207, 0.9647812), (208, 0.9647812), (209, 0.9647812), (210, 0.9647812), (211, 0.9647812), (212, 0.9647812), (213, 0.9647812), (214, 0.9647812), (215, 0.9647812), (216, 0.9647812), (217, 0.9647812), (218, 0.9647812), (219, 0.9647812), (220, 0.9647812), (221, 0.9647812), (222, 0.9647812), (223, 0.9647812), (224, 0.9647812), (225, 0.9647812), (226, 0.9647812), (227, 0.9647812), (228, 0.9647812), (229, 0.9647812), (230, 0.9647812), (231, 0.9647812), (232, 0.9647812), (233, 0.9647812), (234, 0.9647812), (235, 0.9647812), (236, 0.9647812), (237, 0.9647812), (238, 0.9647812), (239, 0.9647812), (240, 0.9647812), (241, 0.9647812), (242, 0.9647812), (243, 0.9647812), (244, 0.9647812), (245, 0.9647812), (246, 0.9647812), (247, 0.9647812), (248, 0.9647812), (249, 0.9647812), (250, 0.9647812), (251, 0.9647812), (252, 0.9647812), (253, 0.9647812), (254, 0.9647812), (255, 0.9647812), (256, 0.9647812), (257, 0.9647812), (258, 0.9647812), (259, 0.9647812), (260, 0.9647812), (261, 0.9647812), (262, 0.9647812), (263, 0.9647812), (264, 0.9647812), (265, 0.9647812), (266, 0.9647812), (267, 0.9647812), (268, 0.9647812), (269, 0.9647812), (270, 0.9647812), (271, 0.9647812), (272, 0.9647812), (273, 0.9647812), (274, 0.9647812), (275, 0.9647812), (276, 0.9647812), (277, 0.9647812), (278, 0.9647812), (279, 0.9647812), (280, 0.9647812), (281, 0.9647812), (282, 0.9647812), (283, 0.9647812), (284, 0.9647812), (285, 0.9647812), (286, 0.9647812), (287, 0.9647812), (288, 0.9647812), (289, 0.9647812), (290, 0.9647812), (291, 0.9647812), (292, 0.9647812), (293, 0.9647812), (294, 0.9647812), (295, 0.9647812), (296, 0.9647812), (297, 0.9647812), (298, 0.9647812), (299, 0.9647812), (300, 0.9647812), (301, 0.9647812), (302, 0.9647812), (303, 0.9647812), (304, 0.9647812), (305, 0.9647812), (306, 0.9647812), (307, 0.9647812), (308, 0.9647812), (309, 0.9647812), (310, 0.9647812), (311, 0.9647812), (312, 0.9647812), (313, 0.9647812), (314, 0.9647812), (315, 0.9647812), (316, 0.9647812), (317, 0.9647812), (318, 0.9647812), (319, 0.9647812), (320, 0.9647812), (321, 0.9647812), (322, 0.9647812), (323, 0.9647812), (324, 0.9647812), (325, 0.9647812), (326, 0.9647812), (327, 0.9647812), (328, 0.9647812), (329, 0.9647812), (330, 0.9647812), (331, 0.9647812), (332, 0.9647812), (333, 0.9647812), (334, 0.9647812), (335, 0.9647812), (336, 0.9647812), (337, 0.9647812), (338, 0.9647812), (339, 0.9647812), (340, 0.9647812), (341, 0.9647812), (342, 0.9647812), (343, 0.9647812), (344, 0.9647812), (345, 0.9647812), (346, 0.9647812), (347, 0.9647812), (348, 0.9647812), (349, 0.9647812), (350, 0.9647812), (351, 0.9647812), (352, 0.9647812), (353, 0.9647812), (354, 0.9647812), (355, 0.9647812), (356, 0.9647812), (357, 0.9647812), (358, 0.9647812), (359, 0.9647812), (360, 0.9647812), (361, 0.9647812), (362, 0.9647812), (363, 0.9647812), (364, 0.9647812), (365, 0.9647812), (366, 0.9647812), (367, 0.9647812), (368, 0.9647812), (369, 0.9647812), (370, 0.9647812), (371, 0.9647812), (372, 0.9647812), (373, 0.9647812), (374, 0.9647812), (375, 0.9647812), (376, 0.9647812), (377, 0.9647812), (378, 0.9647812), (379, 0.9647812), (380, 0.9647812), (381, 0.9647812), (382, 0.9647812), (383, 0.9647812), (384, 0.9647812), (385, 0.9647812), (386, 0.9647812), (387, 0.9647812), (388, 0.9647812), (389, 0.9647812), (390, 0.9647812), (391, 0.9647812), (392, 0.9647812), (393, 0.9647812), (394, 0.9647812), (395, 0.9647812), (396, 0.9647812), (397, 0.9647812), (398, 0.9647812), (399, 0.9647812), (400, 0.9647812), (401, 0.9647812), (402, 0.9647812), (403, 0.9647812), (404, 0.9647812), (405, 0.9647812), (406, 0.9647812), (407, 0.9647812), (408, 0.9647812), (409, 0.9647812), (410, 0.9647812), (411, 0.9647812), (412, 0.9647812), (413, 0.9647812), (414, 0.9647812), (415, 0.9647812), (416, 0.9647812), (417, 0.9647812), (418, 0.9647812), (419, 0.9647812), (420, 0.9647812), (421, 0.9647812), (422, 0.9647812), (423, 0.9647812), (424, 0.9647812), (425, 0.9647812), (426, 0.9647812), (427, 0.9647812), (428, 0.9647812), (429, 0.9647812), (430, 0.9647812), (431, 0.9647812), (432, 0.9647812), (433, 0.9647812), (434, 0.9647812), (435, 0.9647812), (436, 0.9647812), (437, 0.9647812), (438, 0.9647812), (439, 0.9647812), (440, 0.9647812), (441, 0.9647812), (442, 0.9647812), (443, 0.9647812), (444, 0.9647812), (445, 0.9647812), (446, 0.9647812), (447, 0.9647812), (448, 0.9647812), (449, 0.9647812), (450, 0.9647812), (451, 0.9647812), (452, 0.9647812), (453, 0.9647812), (454, 0.9647812), (455, 0.9647812), (456, 0.9647812), (457, 0.9647812), (458, 0.9647812), (459, 0.9647812), (460, 0.9647812), (461, 0.9647812), (462, 0.9647812), (463, 0.9647812), (464, 0.9647812), (465, 0.9647812), (466, 0.9647812), (467, 0.9647812), (468, 0.9647812), (469, 0.9647812), (470, 0.9647812), (471, 0.9647812), (472, 0.9647812), (473, 0.9647812), (474, 0.9647812), (475, 0.9647812), (476, 0.9647812), (477, 0.9647812), (478, 0.9647812), (479, 0.9647812), (480, 0.9647812), (481, 0.9647812), (482, 0.9647812), (483, 0.9647812), (484, 0.9647812), (485, 0.9647812), (486, 0.9647812), (487, 0.9647812), (488, 0.9647812), (489, 0.9647812), (490, 0.9647812), (491, 0.9647812), (492, 0.9647812), (493, 0.9647812), (494, 0.9647812), (495, 0.9647812), (496, 0.9647812), (497, 0.9647812), (498, 0.9647812), (499, 0.9647812), (500, 0.9647812), (501, 0.9647812), (502, 0.9647812), (503, 0.9647812), (504, 0.9647812), (505, 0.9647812), (506, 0.9647812), (507, 0.9647812), (508, 0.9647812), (509, 0.9647812), (510, 0.9647812), (511, 0.9647812), (512, 0.9647812), (513, 0.9647812), (514, 0.9647812), (515, 0.9647812), (516, 0.9647812), (517, 0.9647812), (518, 0.9647812), (519, 0.9647812), (520, 0.9647812), (521, 0.9647812), (522, 0.9647812), (523, 0.9647812), (524, 0.9647812), (525, 0.9647812), (526, 0.9647812), (527, 0.9647812), (528, 0.9647812), (529, 0.9647812), (530, 0.9647812), (531, 0.9647812), (532, 0.9647812), (533, 0.9647812), (534, 0.9647812), (535, 0.9647812), (536, 0.9647812), (537, 0.9647812), (538, 0.9647812), (539, 0.9647812), (540, 0.9647812), (541, 0.9647812), (542, 0.9647812), (543, 0.9647812), (544, 0.9647812), (545, 
```