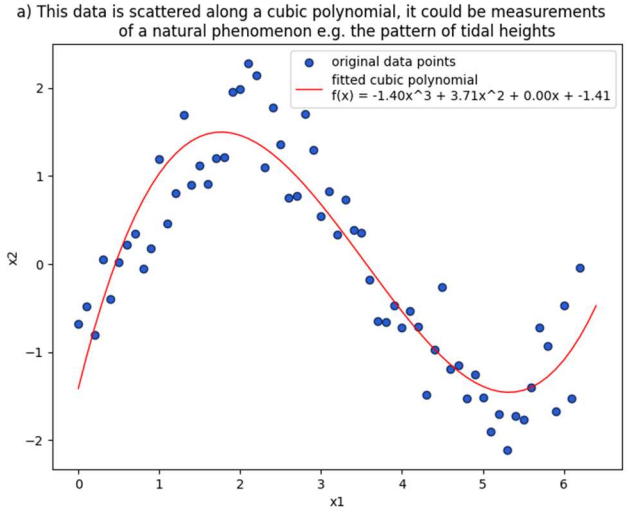
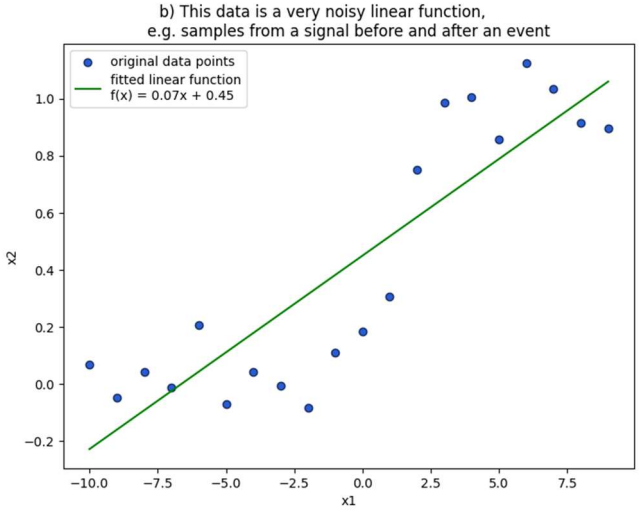
Homework 7



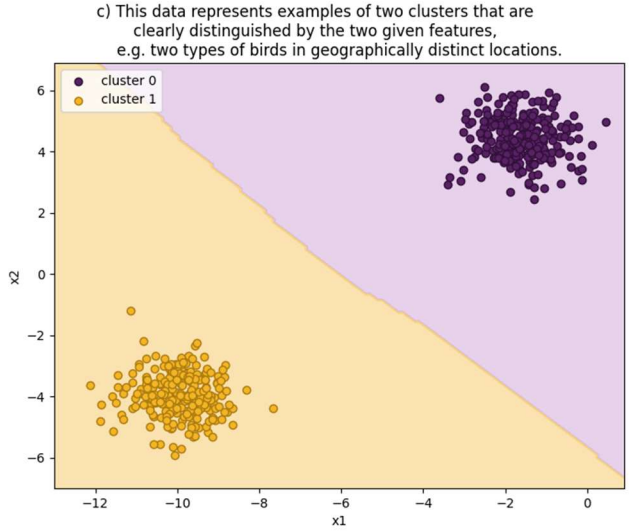

```python
# PolynomialFeatures for data from regression1.csv

X_reg1 = np.array(reg1_df[['x1']])
y_reg1 = reg1_df['x2']

# create polynomial features
poly = PolynomialFeatures(degree=3)
X_poly = poly.fit_transform(X_reg1.reshape((-1,1)))

# perform Linear Regression with a cubic polynomial
poly_reg = LinearRegression()
poly_reg.fit(X_poly, y_reg1)

x_values_poly = np.arange(0, 6.5, 0.1)
polynomial = poly_reg.predict(poly.fit_transform(x_values_poly.reshape((-1, 1))))

# get coefficients of polynomial f(x) = a*x^3 + b*x^2 c*x + d
coefficients = poly_reg.coef_
intercept = poly_reg.intercept_
a, b, c, d = coefficients[2], coefficients[1], coefficients[0], intercept

# plot original data points from regression1 and predicted polyomial
plt.figure(figsize=(8, 6))
plt.scatter(reg1_df['x1'], reg1_df['x2'], color=blue, edgecolor=dark_blue, label='original data points')
plt.plot(x_values_poly, polynomial, '-', color='red', label=f'fitted cubic polynomial\
        \nf(x) = {a:.2f}x^3 + {b:.2f}x^2 + {c:.2f}x + {d:.2f}', linewidth=1)
plt.title(f'a) This data is scattered along a cubic polynomial, it could be measurements \
          \nof a natural phenomenon e.g. the pattern of tidal heights', fontsize=fontsize)
plt.xlabel('x1')
plt.ylabel('x2')
plt.legend(fontsize=fontsize-2)
plt.savefig('./HW7_plots/poly_reg.png')

plt.show()
```

a) I used linear regression to fit a cubic polynomial to the data, thus minimising the sum of least squares between original points and the predicted points on the polynomial. The polynomial is $f(x) = -1{,}4x^3 + 3{,}71x^2 - 1{,}41$, where coefficients are rounded to 2 places after the comma.




```python
# Linear Regression with a linear function for data from regression2.csv

X_reg2, y_reg2 = reg2_df[['x1']], reg2_df['x2']

# perform linear regression with a linear function
lin_reg = LinearRegression()
lin_reg.fit(X_reg2, y_reg2)

line = lin_reg.predict(X_reg2)

# get coefficients of line f(x) = m*x + b
m, b = lin_reg.coef_[0], lin_reg.intercept_
print(m)

# plot the points from regression 2 and the fitted linear function
plt.figure(figsize=(8, 6))
plt.scatter(reg2_df['x1'], reg2_df['x2'], color=blue, edgecolor=dark_blue, label='original data points')
plt.plot(reg2_df['x1'], line, color='green', label=f'fitted linear function\nf(x) = {m:.2f}x + {b:.2f}')
plt.title('b) This data is a very noisy linear function, \
          \ne.g. samples from a signal before and after an event', fontsize=fontsize)
plt.xlabel('x1')
plt.ylabel('x2')
plt.legend(fontsize=fontsize-2)

plt.savefig('.\HW7_plots\lin_reg.png')

plt.show()
```

b) I used linear regression to fit a linear function to the data, thus minimising the sum of least squares between original points and the predicted points on the line given by $f(x) = 0{,}07x + 0{,}45$, where coefficients are rounded to 2 places after the comma.




```python
# kNN for data from classification.csv

# get features and values from data set
X_class = class_df.drop('label', axis=1).values
y_class = class_df['label'].values

# train kNN classifier
k = 3  # number of neighbors
knn = KNeighborsClassifier(n_neighbors=k)
knn.fit(X_class, y_class)

# create a mesh grid for plotting decision boundaries and predict label for each point in grid
xx, yy = np.meshgrid(np.arange(-13, 1, 0.1),
                     np.arange(-7, 7, 0.1))
Z = knn.predict(np.c_[xx.ravel(), yy.ravel()]).reshape(xx.shape)

# define colors for plot
cmap_background = ListedColormap([light_purple, light_yellow])
colors_points = [purple, yellow, dark_purple, dark_yellow]

# create plot
plt.figure(figsize=(8, 6))
plt.contourf(xx, yy, Z, alpha=0.6, cmap=cmap_background)  # plot decision boundary
# plot original data points with cluster-specific labels
for cluster in np.unique(y_class):
    # create mask that identifies which points in y_class belong to the current cluster
    cluster_mask = y_class == cluster
    plt.scatter(X_class[cluster_mask, 0], X_class[cluster_mask, 1], # only choose points (x1,x2) in current cluster
                label=f'cluster {cluster}',
                color=colors_points[cluster], edgecolor=colors_points[cluster + 2])
plt.title(f'c) This data represents examples of two clusters that are\
          \nclearly distinguished by the two given features,\
          \ne.g. two types of birds in geographically distinct locations.', fontsize=fontsize)
plt.xlabel('x1')
plt.ylabel('x2')
plt.legend(fontsize=fontsize-2)
plt.savefig('./HW7_plots/knn_clustering.png')

plt.show()
```

c) I used the kNN classifier with k=3 to classify the data points into two clusters. Since the clusters are very clearly separated, we get an almost linear decision boundary.

Name: Josephina Imhoff
Matriculation No.: 23464631
IdM: jo17wila