Nazwa kierunku:

Informatyka i Ekonometria

Przedmiot:

Inżynierski projekt dyplomowy

Temat pracy:

System do analizy danych „Pynalise” z wykorzystaniem oraz omówieniem technologii Python

Wykonał:

Krzysztof Sendera

Prowadzący: mgr inż. Jarosław Szkoła

Rzeszów 2023

1. Wstęp.

Współczesne rozwiązania informatyczne mają ogromny wpływ na funkcjonowanie wielu branż. Jednym z kluczowych aspektów zarządzania firm jest przetwarzanie danych. W ramach projektu inżynierskiego zdecydowałem się opracować nowoczesny System zarządzania i analizy danych, który umożliwi łatwe i wygodne zarządzanie informacjami. Aplikacja stworzona będzie dla osób zainteresowanych atrybutami oraz statystykami, na podstawie analizy których być może będzie można wysnuć kolejne informacje.

Mechanizmy interakcji użytkownika z aplikacją będą różne, w zależności od roli, którą użytkownik pełni w systemie. Gość będzie mógł przeglądać informacje za pomocą prostego interfejsu graficznego. Użytkownik, dodatkowo będzie miał możliwość wprowadzania zmian w danych oraz dodawania nowych informacji, także za pomocą intuicyjnego interfejsu graficznego. Administrator, poza tymi funkcjami, będzie mógł dodatkowo zarządzać użytkownikami, co również będzie możliwe przez interfejs graficzny.

Prezentacja wyników będzie odbywać się za pomocą różnych widoków, w zależności od potrzeb użytkownika. Gość będzie miał dostęp tylko do widoku informacyjnego, który wyświetli mu dane na temat zwierząt, ich zdjęcia i opisy. Użytkownik będzie miał dodatkowo dostęp do widoku edycji, który pozwoli mu na modyfikowanie danych, dodawanie nowych rekordów oraz usuwanie już istniejących. Administrator będzie miał dodatkowo dostęp do widoku zarządzania Użytkownikami, który umożliwi mu zarządzanie kontami, zmianę ich uprawnień oraz usuwanie.

Potencjalne wąskie gardła mogą pojawić się na poziomie serwera, jeśli będzie on nieodpowiednio skonfigurowany i nie będzie w stanie obsłużyć dużej liczby użytkowników jednocześnie. W takiej sytuacji konieczne będzie zastosowanie odpowiedniego sprzętu lub usług chmurowych, aby zoptymalizować działanie systemu. Innym potencjalnym wąskim gardłem może być sam interfejs graficzny, jeśli nie będzie on intuicyjny i łatwy w obsłudze dla użytkowników. W takim przypadku konieczne będzie przeprowadzenie testów UX i wprowadzenie odpowiednich poprawek.

Celem mojej aplikacji jest przedstawienie użytkownikom informacji o różnych danych. Przykładem reprezentowanym w aplikacji będą dane o zwierzętach, które znajdują się w Zoo w Nottingham. System umożliwi użytkownikom dostęp do różnych informacji, takich jak gatunek zwierzęcia, jego opis, atrybuty, itp. Dzięki temu goście i użytkownicy będą mieli łatwy dostęp do potrzebnych informacji, na podstawie których będą mogli sporządzać wykresy i wyciągać wnioski na temat podobieństw danych gatunków. W pierwszym ekranie użytkownik będzie mógł się zalogować, następnie wyświetli się ekran główny z tabelą z danymi i opcjami które można wybrać by na nich operować. Wykonane obliczenia będą wyświetlać się na ekranie. Dane te będzie można wyeksportować w formie raportu do pliku.

Niniejsza dokumentacja projektu inżynierskiego skupi się na przedstawieniu szczegółów dotyczących tworzenia oraz funkcjonowania systemu wraz z omówieniem technologii wykorzystanych do stworzenia jej. W kolejnych rozdziałach znajdą się informacje

na temat założeń projektowych, opisu funkcjonalności, użytych technologii, architektury, projektowania, tworzenia i testowania systemu.

2. Opis funkcjonalności

Oto lista funkcjonalności, które będą dostępne w aplikacji "Pynalise":

- Logowanie użytkowników - system będzie wymagał, aby użytkownicy się zalogowali przed uzyskaniem dostępu do danych.
- Role użytkowników - aplikacja będzie mieć trzy role: użytkownik, gość i administrator. Każda rola będzie miała dostęp do określonych funkcji.
- Przeglądanie danych - goście i użytkownicy będą mogli przeglądać dane, domyślne dane będą dotyczyć zwierząt w Zoo: gatunek, czy zwierzę poluje, czy składa jaja, czy posiada futro i wiele podobnych..
- Dodawanie danych - użytkownicy będą mogli dodawać nowe dane wczytując je z plików CSV lub JSON.
- Edytowanie danych - użytkownicy będą mogli edytować dane w tabeli.
- Usuwanie danych - użytkownicy będą mogli usuwać dane w tabeli.
- Zarządzanie użytkownikami - administratorzy będą mieli dostęp do funkcji zarządzania użytkownikami, takich jak: dodawanie nowych użytkowników, usuwanie użytkowników, edycja danych osobowych użytkowników.
- Raportowanie - aplikacja będzie miała możliwość generowania raportów dotyczących danych. Raporty będą dostępne w formie plików PDF.
- Statystyki - aplikacja będzie miała możliwość generowania statystyk dotyczących atrybutów danych oraz generowania wykresów dla porównania poszczególnych atrybutów.

Te funkcjonalności pozwolą użytkownikom na przeglądanie, zarządzanie i analizowanie danych w sposób prosty i intuicyjny.

3. Technologie:

Aplikacja będzie desktopowa. W projekcie wykorzystam następujące technologie: Do stworzenia GUI użytkownika wykorzystam technologię języka programowania Python. Bazę danych stworzę na serwerze MySQL. Żeby ukazać lepiej możliwości technologii Python stworzę również stronę internetową używając Framework Django. Użytkownicy będą mogli pobrać program, zgłaszać błędy dotyczące aplikacji oraz przeczytać dokumentację. Wybór technologii jest uzasadniony ze względu na ich popularność, wsparcie społeczności oraz możliwość zintegrowania aplikacji desktopowej z witryną internetową. To podejście może znacząco zwiększyć użyteczność aplikacji i ułatwić jej rozwijanie oraz utrzymanie.

3.1. Wybór typu aplikacji:

Wybór typu aplikacji jest ważną decyzją i zależy od wielu czynników. Podczas wyboru typu aplikacji kierowałem się następującymi argumentami:

- **Wydajność:** Aplikacje desktopowe często oferują wyższą wydajność niż aplikacje webowe lub mobilne, ponieważ działają na komputerze lokalnym, wykorzystując pełną moc obliczeniową urządzenia
- **Funkcjonalność:** Aplikacje desktopowe mają tendencję do oferowania bardziej zaawansowanych funkcji i możliwości, ponieważ nie są ograniczone przez przeglądarkę internetową lub ograniczenia systemu operacyjnego mobilnego
- **Dostęp do zasobów lokalnych:** Aplikacje desktopowe mogą mieć dostęp do zasobów lokalnych, takich jak pliki i dane, co może być ważne w niektórych przypadkach, takich jak praca z dużymi plikami
- **Bezpieczeństwo:** Aplikacje desktopowe mogą być bardziej bezpieczne niż aplikacje webowe, ponieważ dane i operacje są przechowywane na komputerze użytkownika, a nie w chmurze.
- **Brak potrzeby dostępu do internetu:** Aplikacje desktopowe nie zawsze wymagają stałego dostępu do internetu, co może być korzystne w sytuacjach, gdzie łączność internetowa jest niestabilna

Kierując się tymi puntami doszedłem do wniosku, że aplikacja do analizy danych, która może potrzebować dużej mocy obliczeniowej, pamięci, dostępu do plików z urządzenia będzie najlepiej funkcjonować jako aplikacja desktopowa. Jednak istnieją argumenty przemawiające za tym, że aplikacja webowa lub mobilna może być lepszym wyborem, na przykład:

- **Dostępność na wielu urządzeniach:** Aplikacje webowe i mobilne są bardziej dostępne na różnych urządzeniach, co jest przydatne, użytkownik może korzystać z aplikacji na smartfonie, tablecie i komputerze
- **Aktualizacje i utrzymanie:** Aplikacje webowe i mobilne są łatwiejsze do aktualizacji i utrzymania, ponieważ zmiany mogą być wprowadzane centralnie na serwerze, bez konieczności aktualizowania oprogramowania na każdym urządzeniu użytkownika
- **Udostępnianie:** Aplikacje webowe są łatwiejsze do udostępniania, ponieważ nie wymagają instalacji na każdym urządzeniu, a dostęp można uzyskać przez przeglądarkę internetową

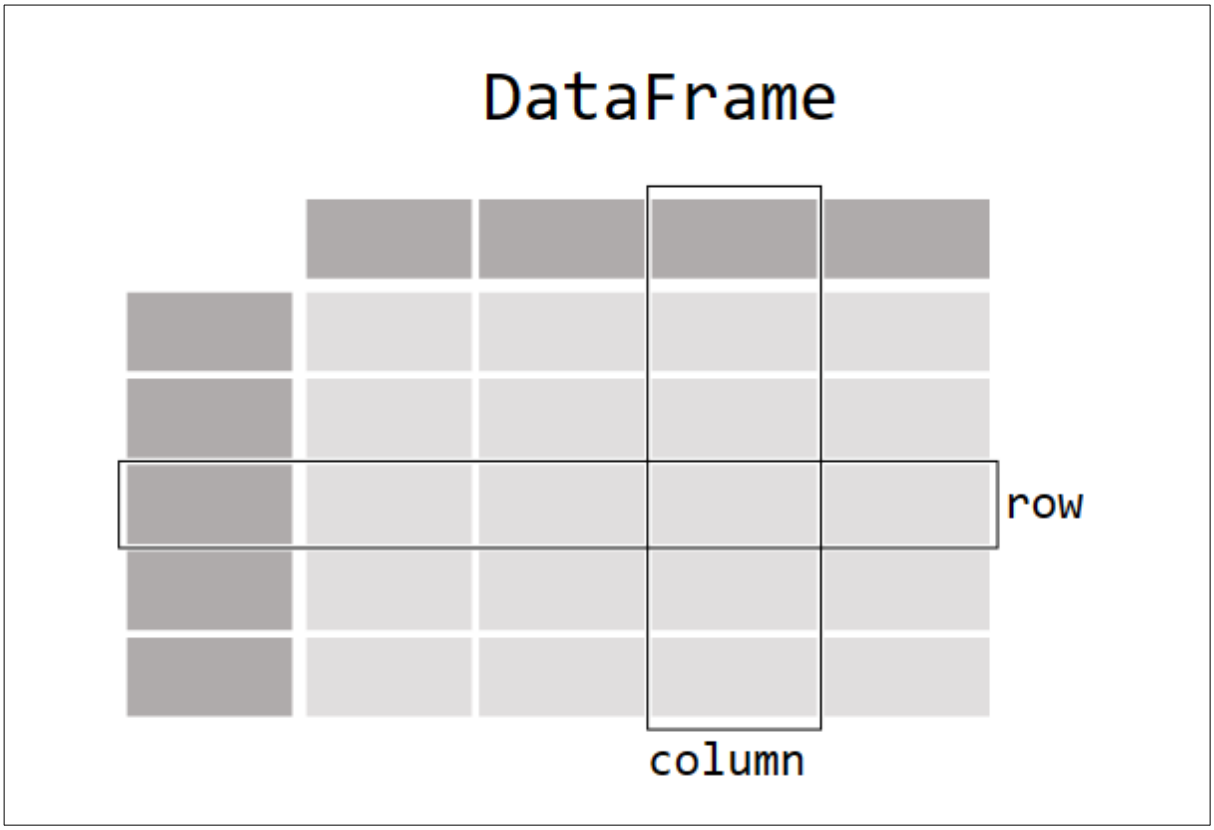
Z tego w względu żeby rozwiązać niektóre z tych problemów postanowiłem stworzyć również prostą stronę internetową, która będzie umożliwiać pobranie aktualnej wersji aplikacji z wielu urządzeń, możliwość zgłoszenia błędu oraz dostęp do dokumentacji systemu.

3.2. Biblioteki Pythona wykorzystane w projekcie:

3.2.1. Pandas

Jest to ogólnodostępna biblioteka, otwarte oprogramowanie wykorzystywane w analizie oraz przetwarzaniu danych. Posiada ona licencję BSD. Jest to elastyczna i permisywna licencja, która daje nieograniczone prawa do użytkowania, kopiowania, modyfikacji i redystrybucji kodu źródłowego Najważniejsze funkcjonalności:

- **DataFrame** – jest to struktura danych i najważniejsze narzędzie oferowane przez to oprogramowanie. Jest to tabela przechowująca dane posiadająca wiersze oraz kolumny do których odpowiednie funkcje dają nam łatwy dostęp oraz możliwość modyfikacji.



Rysunek 1. DataFrame

- Wczytywanie oraz zapisywanie danych jest kompatybilne z różnymi formatami plików oraz źródłami danych (CSV, Excel, JSON, Parquet)
- Pozwala dzielić oraz filtrować dane używając odpowiednio napisanych warunków
- Umożliwia tworzenie nowych kolumn w oparciu o już istniejące
- Współdziała z biblioteką Matplotlib, dzięki czemu można generować wykresy w oparciu o dane zawarte w tabeli
- Zawiera w sobie funkcje umożliwiające obliczenia statystyczne
- Umożliwia pracę z różnymi rodzajami danych, chociażby wspiera pracę z danymi czasowymi, takimi jak daty czy godziny

- Pozwala zachować kontrolę nad czystością danych, poprzez sprawdzanie czy komórki są puste lub wypełnienie ich odpowiednimi wartościami

3.2.2. Matplotlib

To wszechstronna wielofunkcyjna biblioteka służąca do generowania statycznych, ruchomych, interaktywnych wykresów oraz wizualizacji danych. Szeroko wykorzystuje się ją w analizie danych, analizie finansowej, badaniach naukowych czy też w inżynierii. Najważniejsze funkcjonalności Matplotlib:

- Konfigurowalność, czyli możliwość dostosowania wyglądu. Można kontrolować kolorystykę, typy linii, etykiety osi, tytuły, legendy i wiele innych aspektów wykresu
- Wsparcie dla wielu formatów wyjściowych: Umożliwia zapisywanie wykresów w różnych formatach, takich jak PNG, JPEG, SVG, PDF i wiele innych. Dzięki temu można łatwo udostępniać swoje wizualizacje w różnych kontekstach
- Integracja z innymi narzędziami: Matplotlib może być używane w połączeniu z innymi bibliotekami Python, takimi jak NumPy do przetwarzania danych numerycznych, pandas do analizy danych i SciPy do obliczeń naukowych
- Tworzenie różnych rodzajów wykresów:
    - Wykres liniowy (Line Plot): Wykres przedstawiający dane za pomocą linii, które łączą punkty danych
    - Wykres punktowy (Scatter Plot): Wykres przedstawiający punkty danych na płaszczyźnie, gdzie każdy punkt reprezentuje parę wartości (x, y)
    - Wykres słupkowy (Bar Plot): Wykres przedstawiający dane za pomocą słupków o różnej długości, które reprezentują wartości w różnych kategoriach
    - Wykres kołowy (Pie Chart): Wykres przedstawiający dane w formie koła, gdzie każdy kawałek koła reprezentuje procentową część całkowitej wartości
    - Histogram: Wykres przedstawiający rozkład danych numerycznych na podstawie częstotliwości występowania w określonych przedziałach
    - Wykres konturowy (Contour Plot): Wykres przedstawiający dane przestrzenne za pomocą kontur, które reprezentują równość wartości
    - Wykres heatmap (Mapa cieplna): Wykres przedstawiający dane w postaci kolorowych kwadratów na siatce, gdzie kolor reprezentuje wartość w danym punkcie
    - Wykres pudełkowy (Box Plot): Wykres przedstawiający rozkład danych w formie pudełka z wąsami, co pomaga w analizie rozkładu danych i identyfikacji wartości odstających
    - Wykres skrzypcowy (Violin Plot): Wykres łączący wykres pudełkowy z histogramem, umożliwiający lepsze zrozumienie rozkładu danych

- Wykres słupkowy skumulowany (Stacked Bar Plot): Wykres przedstawiający dane za pomocą słupków, w których różne wartości są nakładane na siebie
- Wykres powierzchniowy (3D Surface Plot): Wykres przedstawiający dane przestrzenne w trójwymiarowej przestrzeni za pomocą powierzchni
- Wykres konturowy trójwymiarowy (3D Contour Plot): Wykres przedstawiający dane przestrzenne w trójwymiarowej przestrzeni za pomocą kontur

3.2.3. PyQT6

To złożona biblioteka do tworzenia interfejsu użytkownika dla aplikacji desktopowych opartych na widgetach umożliwiająca obsługę zdarzeń. QT6 to szósta wersja Frameworka stworzonego w języku C++ do tworzenia aplikacji, obsługi ich zdarzeń, komunikacji sieciowej czy wymiany plików. Został on powiązany z językiem Python i w ten sposób powstała biblioteka PyQT6, która umożliwia wykorzystanie kodu źródłowego napisanego domyślnie w C++ w programach tworzonych w technologii Python. PyQT6 posiada licencję GPL v3 (General Public License), która umożliwia swobodne modyfikowanie i rozpowszechnianie tego oporogramowania. Najważniejsze zalety tego narzędzia:

- Obsługa zdarzeń jest jedną z najważniejszych funkcjonalności, dzięki niej możemy kontrolować aplikacje za pomocą kliknięć myszy, naciskając poszczególne przyciski, komponenty, możemy również wprowadzać dane z klawiatury.
- Wieloplatformowość: Aplikacje stworzone w Qt mogą działać na różnych systemach operacyjnych, takich jak Windows, macOS i Linux.
- Qt Designer – jest to narzędzie do tworzenia interfejsu, które bardzo ułatwia projektowanie aplikacji. Można tam układać komponenty metodą przeciągania oraz upuszczania.
- Qt obsługuje wiele języków programowania. Przede wszystkim C++ w którym został napisany oraz Python przy pomocy PyQT6, jednak są również wiązki umożliwiające pracę w innych językach jak Java, Ruby, C#, Perl czy nawet JavaScript
- Kompleksowa dokumentacja i wiele źródeł informacji ze względu na dużą społeczność posługującą się tym narzędziem

3.2.4. scikit-learn

Biblioteka służąca to wykorzystywania algorytmów uczenia maszynowego. To kompleksowe narzędzie wykorzystujące NumPy, matloplib czy SciPy. Posiada licencję BSD podobnie jak pandas. Stworzona została w 2007 roku przez Davida Cournapeau jako Google Summer of Code project. Wielu ludzi o tamtego czasu rozwinęło ten projekt dokładając swoje spostrzeżenia oraz doświadczenie. Najważniejsze cechy tego oprogramowania:

• Wszechstronność: Scikit-learn zawiera bogatą kolekcję algorytmów uczenia maszynowego, takich jak algorytmy klasyfikacji, regresji, grupowania, redukcji wymiarowości, wykrywania

anomalii, nawiasowe, klasteryzacji, uczenie semi-nadzorowane, uczenie wzmacniane, analiza skupień

• Łatwość użycia: Biblioteka jest zaprojektowana w sposób intuicyjny i łatwy do nauki. Oferuje spójne API dla różnych algorytmów, co ułatwia korzystanie z różnych modeli

• Wsparcie dla danych: Scikit-learn obsługuje dane w postaci tabelarycznych struktur danych, takich jak tablice NumPy i ramki danych pandas, co ułatwia pracę z danymi

• Ocena modeli: Biblioteka dostarcza narzędzia do oceny i walidacji modeli, takie jak podział danych na zbiory treningowe i testowe, walidacja krzyżowa, oraz różne metryki oceny wydajności modeli

• Preprocessing danych: Scikit-learn oferuje narzędzia do przetwarzania danych, w tym standaryzację, normalizację, kodowanie zmiennych kategorycznych i wiele innych

• Wizualizacje: Biblioteka umożliwia wizualizację wyników analizy i modeli za pomocą wykresów

• Dokumentacja i społeczność: Scikit-learn posiada obszerną dokumentację, która zawiera przykłady użycia i porady. Ponadto, jest wspierane przez aktywną społeczność użytkowników i programistów

W moim projekcie użyję wybrane algorytmy oraz omówię ich modele matematyczne.

3.2.5. reportlab

To biblioteka, która używana jest do generowania dokumentów oraz grafik w Pythonie. Posiada licencję BSD podobnie jak pandas oraz scikit-learn. Stworzył ją Andy Robinson a następnie rozwijana była przez społeczność. W tym projekcie zostanie wykorzystana do generowania raportów PDF z wynikami obliczeń.

3.2.6. Seaborn

Rozwinięcie biblioteki matplotlib, pozwala na generowanie atrakcyjnych dla oka wykresów oraz grafik statystycznych. Zostanie użyta by lepiej oddać niektóre wykresy. Stworzona została przez Michaela Waskoma w 2013 roku i jest jednym z najpopularniejszych narzędzi wykorzystywanych w tej dziedzinie.

3.2.7. JSON

Jest to standardowy moduł Pythona pozwalający na kodowanie oraz dekodowanie danych w formacie .JSON.

3.3. Środowisko programistyczne

Do stworzenia mojego projektu wybrałem środowisko Pycharm. Jest to bardzo popularne oraz rozbudowane IDE (Integrated Development Enviroment). Oferuje ono wiele fukcjonalności, które sprawiają że praca w nim jest niezwykle skuteczna:

- Inteligentne podpowiedzi – oprogramowanie zapobiega pisania błędów analizując nasz kod na podstawie zaintalowanych domyślnie oraz zewnętrzenie bibliotek
- zaawansowane narzędzia do debugowania pozwalają diagnozować skomplikowane problemy w kodzie i pomagają je rozwiązywać
- jest to narzędzie zintegrowane z Gitem, co pozwala łatwo zabezpieczać kod oraz pracować na nim w grupie
- Wspiera różne frameworki takie jak Django, Flask, Pyramid i wiele innych. Dzięki temu można tworzyć aplikacje webowe i inne projekty oparte na frameworkach.
- Pozwala tworzyć i porządkować duże projekty integrując przy tym różne technologie, obsługuje również inne języki, nie tylko Python
- Rozbudowane mechanizmy wyszukiwania, które pozwalają skutecznie odnaleźć się nawet w bardzo złożonym projekcie

3.4. System kontroli wersji

Oprogramowaniem, które posłuży mi do zapisywania stanu mojego projektu będzie GIT. Jest szeroko stosowany w dziedzinie zarządzania wersjami i kontroli kodu źródłowego w projektach informatycznych. Główną funkcją Gita jest umożliwienie zespołowi programistów pracę na jednym projekcie. Użytkownik tworzy własne repozytorium z kodem do którego umożliwia dostęp i możliwość edycji innym użytkownikom za pomocą poleceń typu PUSH, COMMIT lub PULL. Deweloper może również opublikować kod swojego programu publicznie dzięki czemu różni ludzie mogą go pobrać, rozwinąć, wynaleźć błędy lub po prostu używać. Jest to bardzo dobre narzędzie do śledzenia zmian w projekcie czy tworzenia kopii zapasowych, ponieważ można cofnąć stan projektu do każdej jego aktualizacji. Z tego względu postanowiłem go wykorzystać do pracy z moim projektem, żeby zabezpieczyć swój program.

4. Modele Matematyczne zaimplementowane w aplikacji

Matematyka jest silnie związana z programowaniem. Zanim przełożymy nasz problem na kod warto jest przeanalizować go pod względem matematycznym dzięki czemu nasze rozwiązanie będzie przemyślane i skuteczne. W kolejnych podpunktach przedstawię modele matematyczne, które zostaną zaimplementowane w moim systemie. Przedstawię obliczenia ich złożoności obliczeniowej oraz pamięciowej.

1.1. Klasyfikacja – model algorytmu regresji liniowej
1.2. Klasyfikacja – model algorytmu drzewa decyzyjnego
1.3. Klasteryzacja – model algorytmu k-sąsiadów
1.4. Minimum

Do obliczenia tej wartości wykorzystujemy wzór na minimalną liczbę z dwóch wybranych zmiennych.

$$min = \frac{1}{2}\left(x_1 + x_2 - \left|x_1 - x_2\right|\right)$$

Wzór jest prosty w działaniu. Złożoność obliczeniowa takiego równania to 0(1). Jednak do obliczenia minimalnej wartości dla całej kolumny danych wykorzystamy funkcję min() która wyznacza minimalną wartość z całej tablicy poprzez porównywanie po kolei wartości i zapisanie najmniejszej. Wtedy złożoność obliczeniowa wzrasta do 0(n) ze względu na ilość danych wejściowych, która jest elastyczna.

1.5. Maximum

Analogicznie dla wartości minimalnej obliczamy wartość maksymalną.

$$max = \frac{1}{2}\left(x_1 + x_2 + \left|x_1 - x_2\right|\right)$$

Do obliczenia jej w systemie dla całego zbioru danych wykorzystamy funkcję max() z wbudowanej biblioteki Pythona.

1.6. Odchylenie Standardowe

Oznaczymy sobie tę statystykę jako zmienną σ. Do wyznaczenia tej wartości wykorzystamy również średnią arytmetyczną $\bar{x}$. n to suma elementów w zbiorze. Odchylenie standardowe tych liczb od ich średniej arytmetycznej, to pierwiastek kwadratowy z wariancji.

$$\sigma = \sqrt{\frac{\left(x_1 - \bar{x}\right)^2 + \left(x_2 - \bar{x}\right)^2 + \ldots + \left(x_n - \bar{x}\right)^2}{n}}$$

Do policzenia tego równania wykorzystamy funkcję std() z wbudowanej biblioteki statystycznej Pythona.

1.7. Średnia arytmetyczna

Średnią oznaczymy sobie przez $\bar{x}$. wyliczamy ją obliczając sumę wszystkich elementów od $x_1 do\, x_n$, a następnie tę sumę należy podzielić przez liczbę elementów w zbiorze czyli n. Oto wzór matematyczny reprezentujący ten model:

$$\bar{x} = \frac{x_1 + x_2 + x_3 \ldots + x_n}{n}$$

Złożoność obliczeniowa tego modelu wynosi 0(n), jest to model liniowy. Do obliczania tej funkcji w systemie wykorzystamy wbudowaną funkcję w pythonie mean(), która korzysta z tego wzoru.

1.8. Korelacja

Stworzymy sobie model matematyczny korelacji dla dwóch zmiennych x oraz y. Wykorzystamy do tego wzór na współczynnik korelacji Pearsona. Ten współczynnik mówi nam jaka jest siła i kierunek zależności liniowej pomiędzy naszymi zmiennymi.

Współczynnik r przyjmuje wartości z przedziału [-1,1],

- Im wartość bliższa 1 tym zależność jest silniejsza i dodatnia (jeżeli x rośnie to y rośnie),
- Im wartość bliższa -1 tym zależność jest silniejsza i ujemna (jeżeli x rośnie to y maleje),
- r=0 oznacza brak związku liniowego pomiędzy zmiennymi.

$$r = \frac{\sum (x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum (x_i - \overline{x})^2} \sqrt{\sum (y_i - \overline{y})^2}}$$

Do obliczania korelacji w naszym systemie wykorzystamy funkcję corr(), która oblicza macierz współczynników korelacji Pearsona dla wybranych kolumn w tabeli.

5. Diagramy UML:

5.1. Diagram przypadków użycia:

- Opis Aktorów

Gość - osoba mogąca jedynie wyświetlać informacje w tabeli, może generować wykresy i obliczać funkcje statystyczne,

Użytkownik - może używać funkcji gościa, ale dodatkowo może eksportować dane oraz obliczenia do pliku zewnętrznego. Użytkownik może również modyfikować listę zwierząt.

Administrator – ma dostęp do całej bazy danych zwierząt oraz może zarządzać użytkownikami.

Baza Danych – przechowuje wszystkie informacje na temat użytkowników oraz zwierząt.

- # Dokumentacja przypadków użycia:

Nazwa:	**Rejestracja**
Identyfikator:	P01
Aktorzy:	Gość, Baza Danych
Krótki opis:	Rejestracja użytkownika
Warunki wstępne:	Gość nie ma wcześniej założonego konta w systemie.
Warunki końcowe:	Gość staje się użytkownikiem w systemie po pomyślnym ukończeniu rejestracji.
Główny przepływ zdarzeń:	1. Gość wybiera odpowiedną opcję, aby rozpocząć rejestrację. 2. Gość uzupełnia wszystkie pola, które są obowiązkowe aby ukończyć rejestrację. 3. System dodaje gościa jako nowego użytkownika. 4. Gość może zalogować się na stronie na swoje nowe konto jako użytkownik.
Alternatywne przepływy zdarzeń:	2a. Gość podał nieprawidłowe dane. 3a. Brak połączenia z bazą danych i system nie może dodać nowego użytkownika.
Specjalne wymagania:	Połączenie z bazą danych systemu

Nazwa:	**Generuj Wykres**
Identyfikator:	P02
Aktorzy:	Gość, Użytkownik
Krótki opis:	Generuje wykres
Warunki wstępne:	Wybrane jeden lub dwa atrybuty
Warunki końcowe:	Otwiera się nowe okno z wygenerowanym wykresem
Główny przepływ zdarzeń:	1. Użytkownik wybiera jeden lub dwa atrybuty w oparciu o które aplikacja ma wygenerować wykres. 2. Użytkownik naciska przycisk który generuje wykres. 3. Otwiera się nowe okno wyświetlające wykres. 4. Użytkownik może eksportować, zapisać, analizować lub zamknąć wykres.
Alternatywne przepływy zdarzeń:	2a. Atrybuty wybrane przez użytkownika są takie same co skutkuje wygenerowaniem wykresu w oparciu tylko o jeden atrybut. 3a. Jeśli dane nie będą spełniać wymagań zaimplementowanego wykresu nie zostanie on

	wygenerowany.
Specjalne wymagania:	Specjalna biblioteka do generowania wykresów

Nazwa:	**Oblicz wybraną funkcję**
Identyfikator:	P03
Aktorzy:	Gość, Użytkownik
Krótki opis:	Oblicza funkcję statystyczną dla wybranego atrybutu
Warunki wstępne:	Naciśnięty odpowiedni przycisk
Warunki końcowe:	Wyświetla się obliczony wynik w oknie do wyświetlania obliczeń
Główny przepływ zdarzeń:	1. Użytkownik wybiera dla którego atrybutu ma zostać obliczona funkcja poprzez zaznaczenie kolumny w tabeli. 2. System oblicza daną funkcję. 3. Wynik zostaje wyświetlony w oknie do obliczeń.
Alternatywne przepływy zdarzeń:	2a. Jeśli dane nie będą liczbą tylko ciągiem znaków zostanie wypisany błąd
Specjalne wymagania:	Specjalna biblioteka do obliczania różnych funkcji statystycznych

Nazwa:	**Eksportuj Dane**
Identyfikator:	P04
Aktorzy:	Użytkownik
Krótki opis:	Eksportuje dane do pliku PDF
Warunki wstępne:	Naciśnięty odpowiedni przycisk
Warunki końcowe:	Wyświetla się komunikat o tym że plik został poprawnie wyeksportowany oraz zapisany.
Główny przepływ zdarzeń:	1. Użytkownik decyduje się na eksportację danych i naciska przycisk. 2. Użytkownik wybiera gdzie plik ma zostać zapisany na dysku. 3. Użytkownik zostaje poinformowany o prawdiłowym wyeksportowaniu danych i zapisaniu pliku.
Alternatywne przepływy zdarzeń:	2a. Jeśli użytkownik nie ma miejsca na dysku plik nie zostanie zapisany.
Specjalne wymagania:	Komponent do wyboru miejsca na dysku.

Nazwa:	**Eksportuj Obliczenia**

Identyfikator:	P05
Aktorzy:	Użytkownik
Krótki opis:	Eksportuje obliczenia do pliku PDF
Warunki wstępne:	Naciśnięty odpowiedni przycisk
Warunki końcowe:	Wyświetla się komunikat o tym że plik został poprawnie wyeksportowany oraz zapisany.
Główny przepływ zdarzeń:	1. Użytkownik decyduje się na eksportację wyników i naciska przycisk. 2. Użytkownik wybiera gdzie plik ma zostać zapisany na dysku. 3. Użytkownik zostaje poinformowany o prawidłowym wyeksportowaniu danych i zapisaniu pliku.
Alternatywne przepływy zdarzeń:	2a. Jeśli użytkownik nie ma miejsca na dysku plik nie zostanie zapisany.
Specjalne wymagania:	Komponent do wyboru miejsca na dysku.

Nazwa:	**Edytuj dane**
Identyfikator:	P06
Aktorzy:	Użytkownik, Admistrator
Krótki opis:	Edytuje dane w tabeli
Warunki wstępne:	Naciśnięty odpowiedniej komórki w tabeli
Warunki końcowe:	Dane wyświetlane w tabeli się zmieniają
Główny przepływ zdarzeń:	1. Użytkownik naciska komórkę 2. Użytkownik wypełnia komórkę nową informacją. 3. Użytkownik zatwierdza zmiany. 4. Zmiany przesłane i zapisane są w bazie danych.
Alternatywne przepływy zdarzeń:	3a Jeśli pola zostały wypełnione nieprawidłowymi danymi zostanie wypisany odpowiedni komunikat. 4a. Jeśli nie ma połączenia z serwerem bazy danych wystąpi błąd.
Specjalne wymagania:	Połączenie z bazą danych

Nazwa:	**Wczytaj Dane**
Identyfikator:	P07
Aktorzy:	Użytkownik, Admistrator
Krótki opis:	Wczytuje dane z pliku CSV lub JSON do tabeli
Warunki wstępne:	Naciśnięty odpowiedni przycisk
Warunki końcowe:	Dane zostają dopisane do tabeli.

Główny przepływ zdarzeń:	1. Użytkownik naciska przycisk wczytania danych 2. Użytkownik wypełnia odpowiednie pola. 3. Użytkownik zatwierdza decyzje. 4. Dane przesłane i zapisane są w bazie danych.
Alternatywne przepływy zdarzeń:	3a Jeśli pola zostały wypełnione nieprawidłowymi danymi zostanie wypisany odpowiedni komunikat. 4a. Jeśli nie ma połączenia z serwerem bazy danych wystąpi błąd.
Specjalne wymagania:	Połączenie z bazą danych

Nazwa:	**Usuń dane**
Identyfikator:	P08
Aktorzy:	Użytkownik, Admistrator
Krótki opis:	Usuwa dane w tabeli i w bazie danych
Warunki wstępne:	Naciśnięty odpowiedni przycisk
Warunki końcowe:	Odpowiedni wiersz w tabeli znika.
Główny przepływ zdarzeń:	1. Użytkownik wybiera wiersz w tabeli z danym zwierzęciem. 2. Użytkownik naciska przycisk usuwania zwierzęcia 3. Odpowiedni wiersz znika z tabeli. 4. Analogiczne zmiany zachodzą w bazie danych.
Alternatywne przepływy zdarzeń:	4a. Jeśli nie ma połączenia z serwerem bazy danych wystąpi błąd.
Specjalne wymagania:	Połączenie z bazą danych

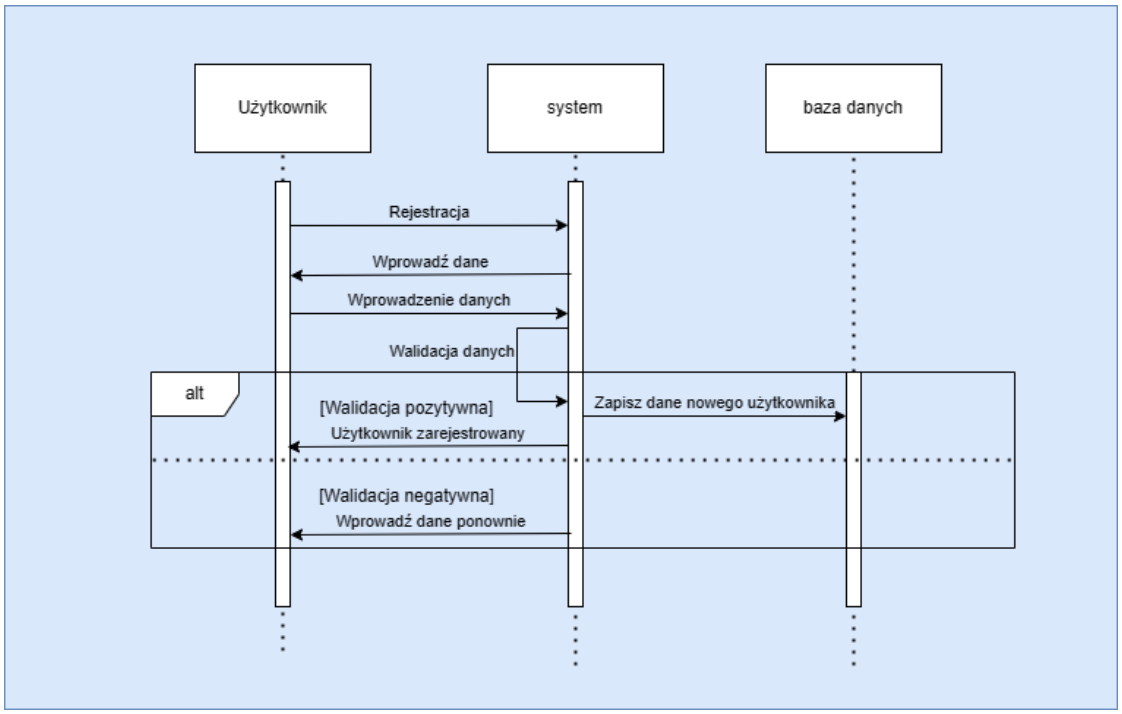
Nazwa:	**Dodaj Użytkownika**
Identyfikator:	P09
Aktorzy:	Administrator
Krótki opis:	Dodaje Użytkownika do bazy danych.
Warunki wstępne:	Naciśnięty odpowiedni przycisk
Warunki końcowe:	Nowy użytkownik zostaje dodany w tabeli wyświetlającej ich.
Główny przepływ zdarzeń:	1. Administrator naciska przycisk dodania użytkownika. 2. Wypełnia odpowiednio pola. 3. Zatwierdza dane. 4. Dane zostają dodane do tabeli oraz do bazy danych.
Alternatywne przepływy zdarzeń:	3a Jeśli pola zostały wypełnione nieprawidłowymi danymi zostanie wypisany odpowiedni komunikat. 4a. Jeśli nie ma połączenia z serwerem bazy danych wystąpi błąd.

Specjalne wymagania:	Połączenie z bazą danych

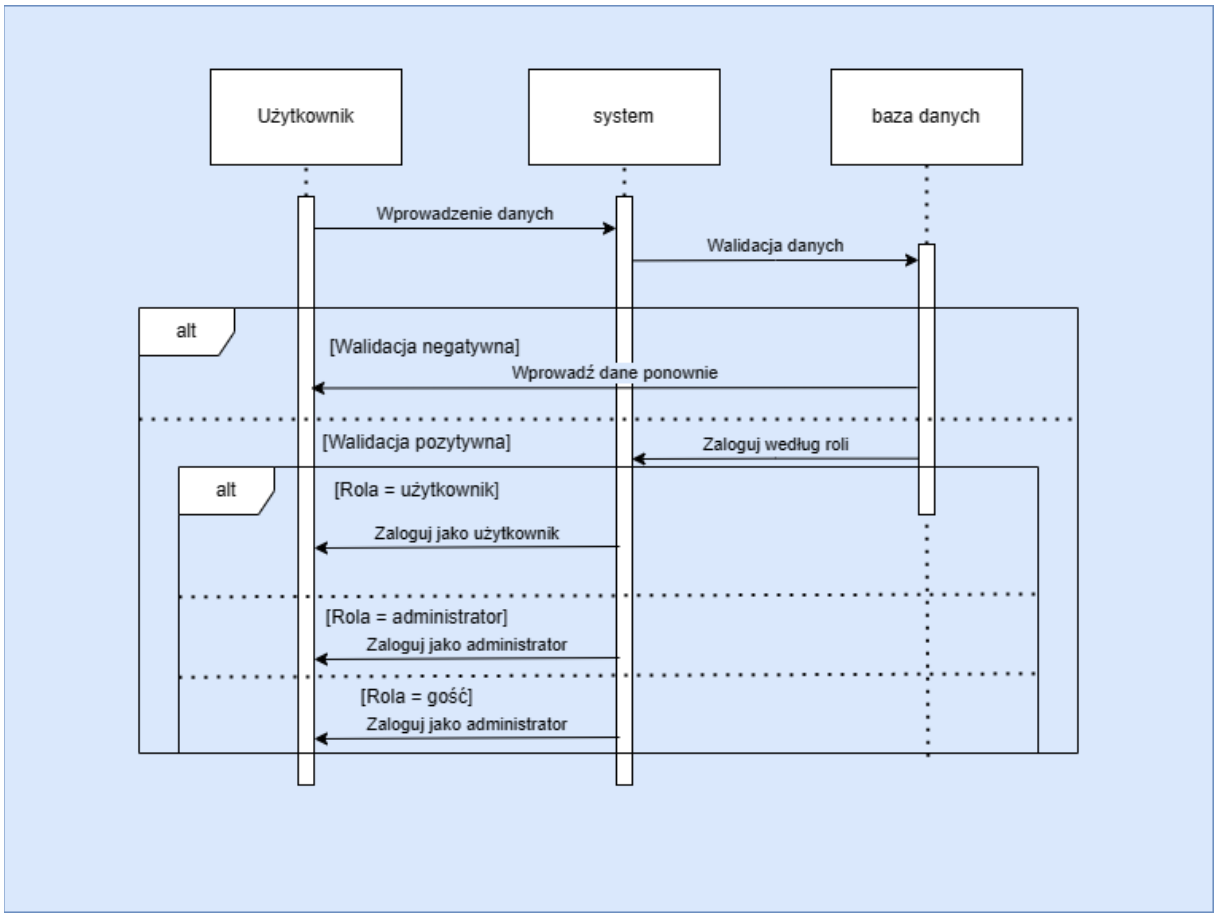
Nazwa:	**Usuń Użytkownika**
Identyfikator:	P10
Aktorzy:	Administrator
Krótki opis:	Usuwa użytkownika w tabeli i w bazie danych
Warunki wstępne:	Naciśnięty odpowiedni przycisk
Warunki końcowe:	Odpowiedni wiersz w tabeli znika.
Główny przepływ zdarzeń:	1. Administrator wybiera wiersz w tabeli z danym Użytkownikiem. 2. Administrator naciska przycisk usuwania zwierzęcia 3. Odpowiedni wiersz znika z tabeli. 4. Analogiczne zmiany zachodzą w bazie danych.
Alternatywne przepływy zdarzeń:	4a. Jeśli nie ma połączenia z serwerem bazy danych wystąpi błąd.
Specjalne wymagania:	Połączenie z bazą danych

5.2. Diagramy sekwencji:

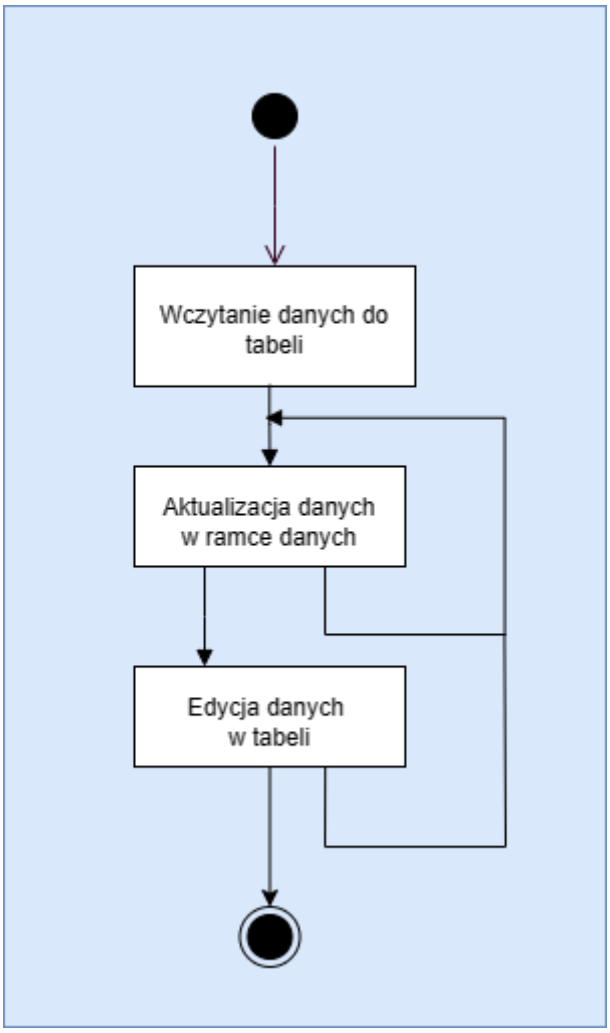
5.2.1. Diagram sekwencji dla Rejestracji nowego użytkownika



5.2.2. Diagram sekwencji dla Logowania do systemu



5.3. Diagram Stanów Tabeli



6. Proces tworzenia aplikacji

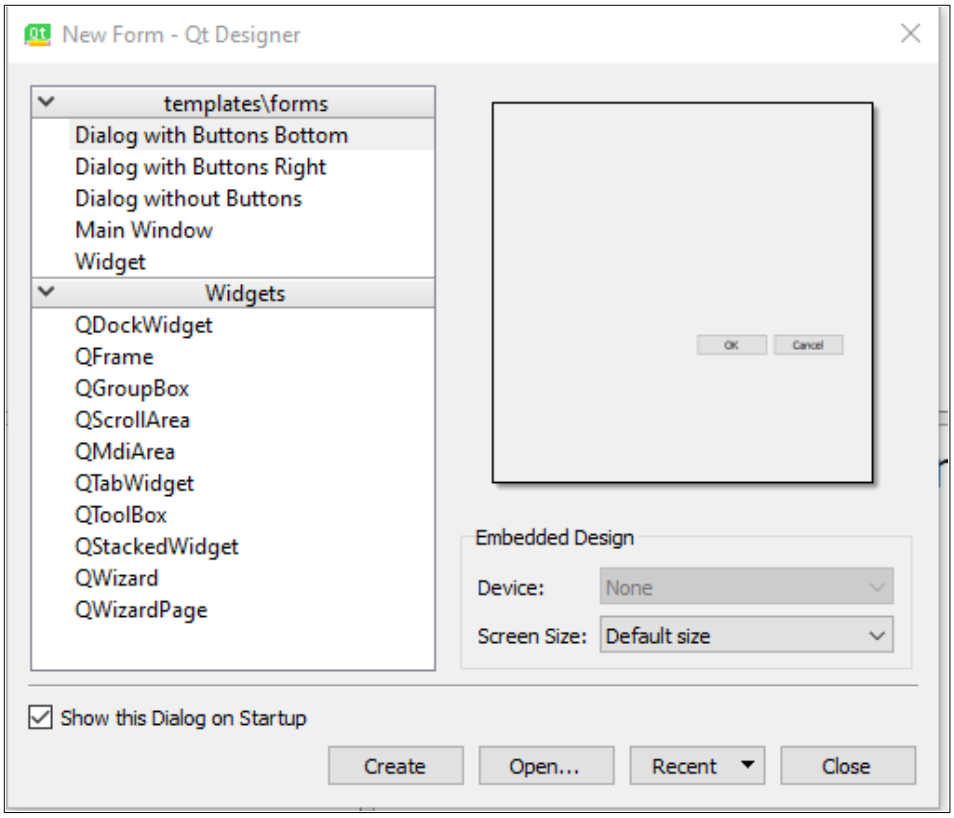
6.1. Projektowanie User Interface

6.1.1. Przegląd dostępnych technologii na rynku.

Zacząłem szukać informacji o tym w jaki sposób najlepiej stworzyć interfejs użytkownika. Przejrzałem dostępne na rynku technologie. Pierwszą biblioteką do tworzenia UI na którą warto zwrócić uwagę jest Tkinter. Jest to standardowa technologia do tworzenia aplikacji w pythonie. Jej atutami jest prostota oraz wieloplatformowość, ponieważ działa ona różnych systemach operacyjnych. Kolejną biblioteką wartą uwagi jest Kivy. Jest to framework, który oprócz tego że pozwala tworzyć UI dla użytkowników windowsa czy Linusa pozwala również na programowanie interfejsów aplikacji mobilnych, animacji, elementów dotykowych czyli programów na system Adroid czy IOS.

6.1.2. Wybór PyQT6

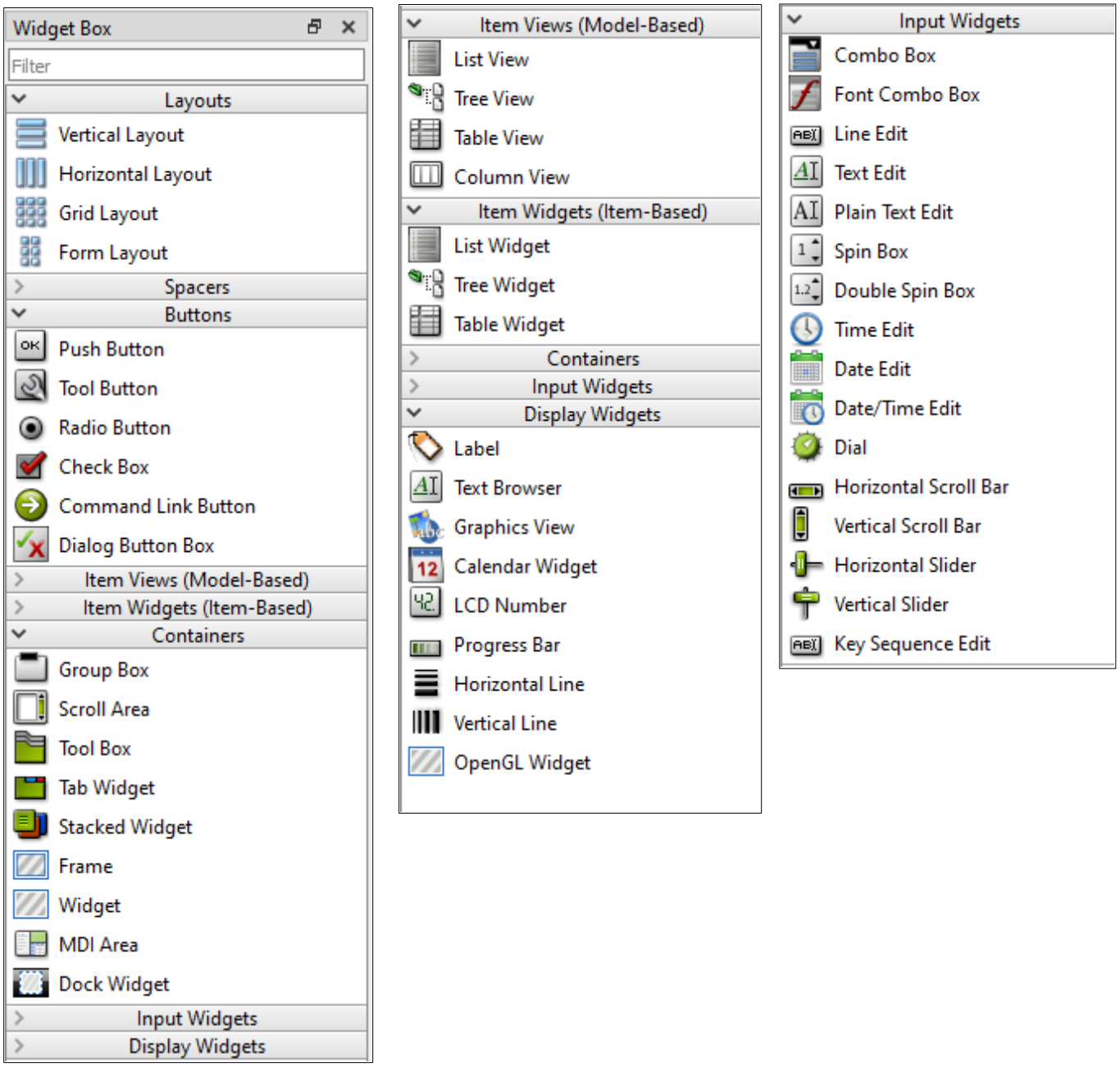
Ja jednak postanowiłem nie skorzystać z żadnej wyżej opisanych możliwości. Technologia którą wybrałem to PyQT6, która jest lepiej skonstruowana do tworzenia złożonych projektów niż inne technologie. Dodatkowo ma estetyczny wygląd. Funkcjonuje lepiej w kontekście wielu platform niż Tkinter. Lepiej też przystosowana jest do tworzenia aplikacji desktopowych niż Kivy, która skoncentrowana jest na aplikacjach mobilnych. Kolejnym aspektem który przemawia za używaniem PyQT6 jest QT Designer. Jest to narzędzie służące do projektowania aplikacji za pomocą przeciągania różnych komponentów w oknie projektowania aplikacji. Qt Designer jest niezależny od platformy i języka programowania. Nie tworzy kodu w żadnym konkretnym języku, ale tworzy pliki .ui. Pliki te mają rozszerzenie .XML ze szczegółowymi opisami generowania GUI opartych na Qt. Ja preferuję konwersję plików .ui na kod Pythona, ze względu na ujednolicenie kodu aplikacji co zmniejsza szansę na wystąpienie nieprzewidzianych błędów podczas eksportowania aplikacji użytkowej.



Omówimy teraz architekturę dostępnych w tej aplikacji okien. Dostępne do wyboru są trzy szablony okien: okno główne, okno dialogowe oraz widżet. Okno główne służy do wyświetlania najważniejszego widoku aplikacji. Jest to ten komponent, który posiada główne funkcjonalności całego programu, domyślnie posiada już pasek zadań, który można wykorzystać do własnych funkcji. Okna dialogowe są dodatkowe. Służą do podejmowania decyzji np. Jeśli chcemy zapytać użytkownika o dodatkowe informacje, możemy zaprojektować okno dialogowe w którym będzie je wprowadzał. Natomiast widżety są tak naprawdę wszystkim w QT. Wszystkie okna dialogowe, okna główne, komponenty, przyciski są tak naprawdę rozbudowanymi widżetami. Szablon widżetu pozwala nam zaprojektować własny, od podstaw, decydując o każdym jego aspekcie. Do wyboru mamy również szereg gotowych widżetów proponowanych przez oprogramowanie.

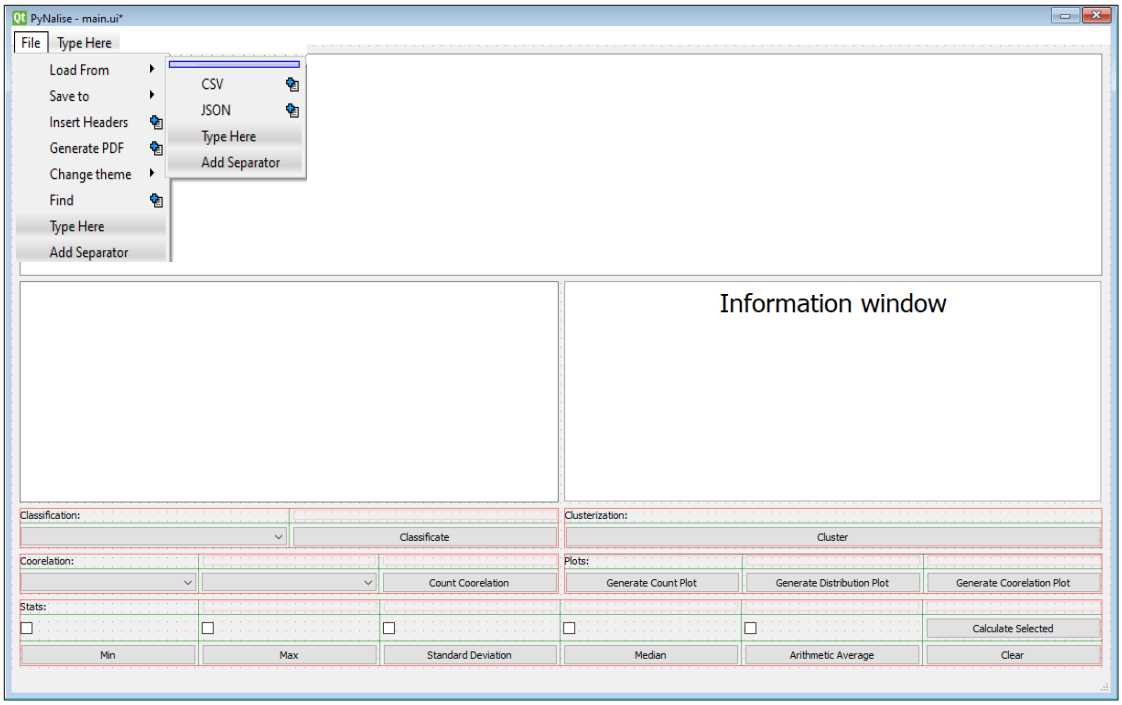
6.1.3. Projektowanie okien do Pyanalise.

Przedstawię teraz mój interfejs oraz proces tworzenia go krok po kroku używając zasobów opisanych powyżej. Pierwszym krokiem było zaplanowanie ilości okien w aplikacji. Przewidziałem trzy okna. Główne okno w którym wyświetlona zostanie tabela, przyciski oraz obliczenia, widżet logowania oraz widżet rejestracji. Kolejnym aspektem QT na który należało zwrócić uwagę jest „Widget Box”. W tej części oprogramowania do wykorzystania mamy gotowe komponenty, które posłużą nam do obsługi zdarzeń w naszej aplikacji, między innymi: kontenery, rozmieszczenia, przyciski, widoki do wyświetlania oraz wprowadzania danych.

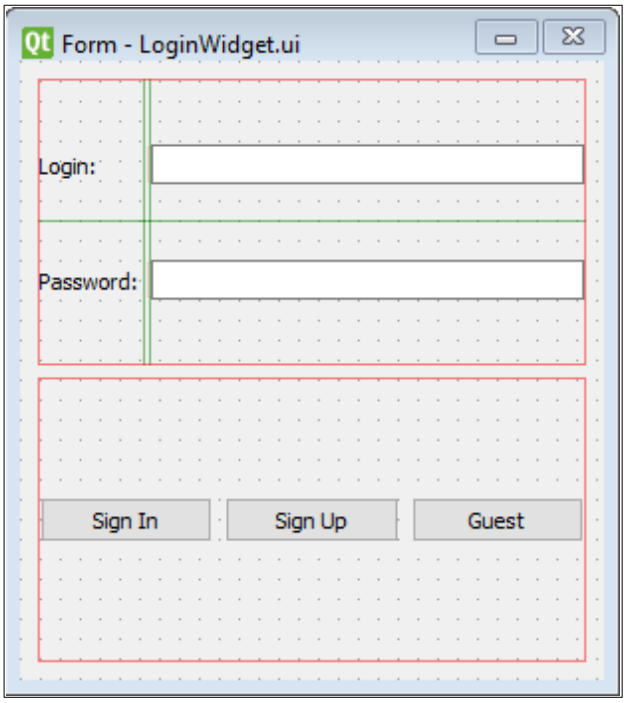


Pierwszym widżetem który użyję do projektu głównego okna jest rozmieszczenie GridLayout. Pozwala ono by program domyślnie ustawiał elementy w równej odległości, co ułatwia porządkowanie i poprawia estetykę programu. Widżety które użyję do stworzenia głównego okna mojego programu to:

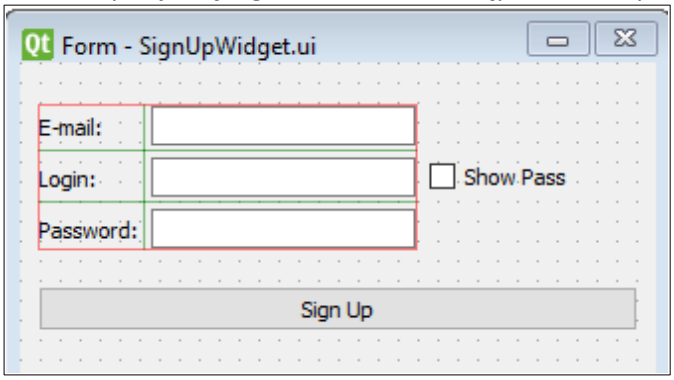
- TableView czyli widok tabeli z danymi.
- TextBrowser użyję do wyświetlania obliczeń oraz informacji na temat funkcji programu.
- Push Button wykorzystam do obsługi zdarzeń dla danych. Będzie to szereg przycisków służących do uruchamiania funkcji obliczających różne algorytmy.
- Label będzie służył do podpisania danych grup z przyciskami.



Do Stworzenia tego okna wykorzystałem pięć GridLayoutów. Do każdego z nich wstawiłem różne przyciski. Do layoutu Classification wstawiłem ComboBox do wyboru klasyfikatora do mojego algorytmu oraz przycisk który będzie uruchamiał zdarzenie wywołujące funkcję klasyfikacji. Do layoutu Clusterisation wstawiłem przycisk wywołujący funkcję która będzie uruchamiać algorytm k-sąsiadów grupujący dane kolumn zaznaczonych w tabeli. Do layoutu Coorelation wstawiłem dwa ComboBoxy w których będzie mogli wybrać dwa atrybuty do obliczenia korelacji pomiędzy nimi. W layoucie podpisanym jako Plots wstawiłem trzy przyciski do generowania trzech różnych wykresów, każdy z nich będzie generował wykres dla zaznaczonych kolumn danych w tabeli. W piątym layoucie o nazwie Stats wstawione jest 5 przycisków które wywołują funkcje obliczające dane statystyczne dla zaznaczonych kolumn. Będzie możliwość zaznaczenia kilku z nich za pomocą checkboxów i policzenia zaznaczonych funkcji statystycznych za pomocą jednego przycisku Calculate Selected. Jest tu również przycisk Clear który będzie czyścił okno wyświetlające obliczenia. Na środku są dwa TextBrowsery . Jeden do wyświetlania obliczeń i drugi do wyświetlania informacji o przyciskach. U góry widać TableView, natomiast w lewym górnym rogu jest pasek menu z dodatkowymi funkcjami. Będzie tam wczytywanie oraz zapisywanie plików dla różnych użytkowników, wpisywanie nagłówków do tabeli, generowanie raportu PDF, zmiana motywu aplikacji oraz wyszukiwanie danych w tabeli. Następnie stworzyłem okno logowania. Jest ono mniej złożone niż główne okno aplikacji.



Najpierw wstawiłem dwa GridLayouty. W dolnym uporządkowałem przyciski uruchamiające funkcje logowania jako użytkownik lub administrator, logowania jako gość oraz rejestracji. Do górnego wstawiłem LineEdity w które wpisuje się login oraz hasło. Następnie stworzyłem okno rejestracji.



Wstawiłem gridlayout w którym umieściłem LinedEdity do wprowadzania danych niezbędnych do rejestracji oraz umieściłem swobodnie checkbox do pokazywania hasła oraz przycisk wywołujący funkcję rejestracji. Okna zapisałem w rozszerzeniu .ui wraz z innymi plikami projektu.

6.1.4. Konwertowanie plików .ui do Python

Aby ułatwić sobie pracę przy kodowaniu okien postanowiłem zmienić reprezentację plików na .py.

Do konwersji plików .ui które reprezentują okna za pomocą języka XML wykorzystałem pyuic6. Jest to narzędzie wiersza poleceń dostarczane wraz z biblioteką PyQt6, służące do konwertowania plików interfejsu użytkownika (UI) utworzonych w programie Qt Designer (lub innym narzędziu do projektowania interfejsów opartym na Qt) na kod źródłowy w języku Pythona. Ten wygenerowany kod Pythona będzie następnie używany do tworzenia interfejsu użytkownika w aplikacji.

Aby przekonwertować plik należy wykonać polecenie:

```
pyuic6 -x main.ui -o main.py
```

-x oznacza że plik może zawierać dodatkowy kod który wykona się podczas importowania pliku, natomiast -o oznacza output, określa do jakiego pliku ma zostać zapisany wygenerowany kod.

6.2. Kodowanie komunikacji między oknami.

Mając gotowe pliki okien interfejsu pierwszą rzeczą jaką trzeba zakodować jest przechodzenie między danymi oknami. Pierwszą podstawową rzeczą, którą trzeba zrobić jest stworzenie instancji klasy QApplication z modułu QtWidgets z biblioteki PyQt6.

```
app = QtWidgets.Qapplication(sys.argv)
```

- QtWidgets: Jest to moduł w bibliotece PyQt6, który zawiera klasy i funkcje związane z elementami interfejsu użytkownika (UI) w Qt.
- QApplication: Jest to główna klasa reprezentująca aplikację Qt. Tworzenie obiektu tej klasy jest konieczne, aby uruchomić aplikację PyQt. QApplication zarządza zasobami systemowymi i obsługuje zdarzenia, takie jak kliknięcia myszy, klawisze klawiatury itp.
- sys.argv: Jest to lista argumentów przekazywanych do programu wierszem poleceń. W przypadku aplikacji PyQt, sys.argv zawiera ścieżkę do pliku wykonywalnego oraz inne argumenty, które mogą być przekazywane do aplikacji przy jej uruchamianiu.

Należy zaimportować do klasy głównej klasy okien z ich plików.

```
from main_ui import Ui_MainWindow
from LoginWidget import LoginWidget
from SignUpWidget import SignUpWidget
```

Pozwoli to stworzyć obiekty tych klas które będą naszymi oknami wyświetlanymi w czasie działania aplikacji.

```
sign_up_window = SignUpWidget()
login_window = LoginWidget(sign_up_window)
login_window.show()
sys.exit(app.exec())
----------------------------------
main_window = MainWindow()
main_window.show()  # Pokaż główne okno
```

Nie jest to cały kod odpowiedzialny za komunikację okien. Jest to wybrana część kodu która ukazuje tworzenie obiektów okien. Każde okno ma swoją klasę w której okodawne są komponenty odpowiedzialne za funkcjonalności danego okna oraz specjalna metoda `def __init__` która uruchamia część kodu podczas tworzenia danego obiektu, jest konstruktorem, będzie nam ona szczególnie przydatna w kolejnych etapach kodowania naszych okien.

6.3. Kodowanie przycisków odpowiedzialnych za funkcjonalności systemu.

6.3.1. Programowanie metod inicjalizujących klasy okien

W pierwszej kolejności musimy uzupełnić metodę init dla klasy głównego okna. Klasa musi dostać informacje kto zalogował się do systemu żeby określić dostępność

funkcjonalności. Jeśli zaloguje się gość część funkcjonalności zostanie ograniczona w tym segmencie kodu poprzez nieuwidocznienie poszczególnych elementów okna głównego. Kolejnym istotnym aspektem jest stworzenie obiektu DataFrame, przypisanie danych do tabeli poprzez stworzenie modelu danych oraz stworzenie obiektu DataFrame do którego również wprowadza się dane, ponieważ jest to struktura danych która oferuje dużo funkcji do operowania na danych oraz jest ona kompatybilna z wieloma innymi bibliotekami. Wstawiamy również odpowiednie dane z nagłówków do comboboxów. Uruchamiamy wątek odpowiedzialny za przesyłanie edytowanych danych z tabeli do DataFrame. Następnie programujemy zdarzenia przycisków oraz menu.

```
# obsługa paska menu
self.actionCSV_load.triggered.connect(
    lambda: Functions.load_CSV_file(self.comboBoxAtrybut1, self.comboBoxAtrybut2, self.comboBoxAtrybutClass,
                                    self.tableView))
self.actionCSV_save.triggered.connect(lambda: Functions.save_to_CSV())
self.actionJSON_load.triggered.connect(
    lambda: Functions.load_JSON_file(self.comboBoxAtrybut1, self.comboBoxAtrybut2, self.comboBoxAtrybutClass,
                                    self.tableView))
self.actionJSON_save.triggered.connect(lambda: Functions.save_to_JSON())
self.actionWprowadzNagl.triggered.connect(
    lambda: Functions.add_headers_manually(self.comboBoxAtrybut1, self.comboBoxAtrybut2,
                                           self.comboBoxAtrybutClass, self.tableView))
self.actionGenerateResultsPDF.triggered.connect(lambda: Functions.generate_pdf(self.textBrowser))
self.actionCiemny.triggered.connect(lambda: Functions.change_to_darkmode(self, self.tableView, self.textBrowserInfo))
self.actionJasny.triggered.connect(lambda: Functions.change_to_lightmode(self, self.tableView))
self.actionWyszukaj.triggered.connect(lambda: self.search())
```

Każde zdarzenie zaprogramowane jest w następującym schemacie
self.nazwa_akcji.triggered.connect(lambda:funkcja(argumenty))
self oznacza dostęp do obiektu który został zainicjalizowany w tym kodzie, w tym przypadku będzie to obiekt okna głównego. Okno główne ma menu, każdy przycisk menu ma własną nazwę np. actionCSV_load to przycisk w menu do wczytywania danych z pliku CSV. Słowo kluczowe „triggered” oznacza, że zostało uruchomione zdarzenie np. Naciśnięcie przycisku w menu. Słowo kluczowee „connect” określa metodę która służy do łączenia zdarzenia danego elementu interfejsu z odpowiednią funkcją w skrypcie. Następnie wywoływana jest funkcja anonimowa za pomocą słowa kluczowego lambda. Zanim zacząłem używać tych funkcji natrafiłem na problem, ponieważ funkcje uruchamiały się podczas inicjalizacji przycisków w aplikacji, a nie po naciśnięciu przycisku co było błędem. Udało mi się rozwiązać ten błąd za pomocą opóźnienia wywoływania funkcji przez użycie lambdy. Niektóre funkcje przyjmują jako argumenty obiekty komponentów interfejsu graficznego, co pozwala danym funkcjom je modyfikować.

```
# przyciski statystyczne: Obsługa:
self.pushButtonMinimum.clicked.connect(lambda: Functions.calculate_minimum(self.textBrowser, self.tableView))
self.pushButtonMaximum.clicked.connect(lambda: Functions.calculate_maximum(self.textBrowser, self.tableView))
self.pushButtonClear.clicked.connect(lambda: self.textBrowser.clear())
self.pushButtonMean.clicked.connect(lambda: Functions.calculate_mean(self.textBrowser, self.tableView))
self.pushButtonStd.clicked.connect(lambda: Functions.calculate_std(self.textBrowser, self.tableView))
self.pushButtonMedian.clicked.connect(lambda: Functions.calculate_median(self.textBrowser, self.tableView))
self.pushButtonDystrybucja.clicked.connect(lambda: Functions.generate_distribution_plot(self.tableView))
self.pushButtonKoorelacja.clicked.connect(
    lambda: Functions.calculate_coorelation(self.comboBoxAtrybut1, self.comboBoxAtrybut2, self.textBrowser,
                                            self.tableView))
self.pushButtonCalcChecked.clicked.connect(
    lambda: Functions.calculate_checked_stats(self.checkBoxMin, self.checkBoxMax, self.checkBoxStd, self.checkBoxMdn,
                                              self.checkBoxMean, self.textBrowser, self.tableView))
self.pushButtonHeatmap.clicked.connect(lambda: Functions.generate_correlation_heatmap(self.tableView))
self.pushButtonClass.clicked.connect(
    lambda: Functions.classificate_selected_data(self.comboBoxAtrybutClass, self.textBrowser))
self.pushButtonCluster.clicked.connect(
    lambda: Functions.cluster_selected_dataX())
self.pushButtonPorownaj.clicked.connect(lambda: Functions.generate_comparison_plot(self.tableView))
```

W analogiczny sposób zakodowałem przyciski odpowiedzialne za obliczenia statystyczne oraz algorytmy uczenia maszynowego. Następnie trzeba zaprogramować podpięte funkcje, ponieważ na tym etapie tworzenia aplikacji są one jeszcze puste, więc po wywołaniu ich nic się nie stanie.

6.3.2. Kodowanie funkcji odpowiedzialnych za obliczenia matematyczne

- Funkcja wyznaczająca minimum dla danej kolumny

```
2 usages (1 dynamic)  Krzysztof Sendera +1
def calculate_minimum(QtTextBrowser, QtTableView): #git
    try:
        for column in selected_columns:
            # Oblicz minimum z wybranej kolumny
            minimum = df[column].min()

            # Konwertuj wartość na string
            minimum_str = str(minimum)

            # Pobierz tekst nagłówka kolumny z QTableView
            header_text = QtTableView.model().headerData(column, Qt.Orientation.Horizontal)

            # Wyświetl wartość w QTextBrowser
            QtTextBrowser.append(f"Minimum from column {header_text} is: {minimum_str}")
    except Exception as e:
        QtTextBrowser.append(f"Error occurred: {str(e)}")
```

Nazwa funkcji to calculate_minimum, posiada ona dwa argumenty QtTextBrowser oraz QtTableView co oznacza że przyjmuje ona do siebie dwa obiekty widżetów: Tabeli oraz ekranu do wyświetlania obliczeń. Do obliczenia wartości wykorzystujemy funkcję .min(), która jest wbudowana w Pythonie i można ją użyć na każdej iterowalnej strukturze danych. W tym przypadku oblicza się ją na podstawie zanaczone kolumny w tabeli i zapisuje do zmiennej minimum. Następnie wartość konwertowana jest jako ciąg znaków tekstowych i wyświetlana w obiekcie ekranu za pomocą metody append(). Dodatkowo cała funkcja zabezpieczona jest by po

wystąpieniu błędu nie zawiesiła programu tylko wypisał go na ekranie.

- Funkcja wyznaczająca maksimum dla danej kolumny

```
def calculate_maximum(QtTextBrowser, QtTableView): #git
    try:
        for column in selected_columns:
            # Oblicz maksimum z wybranej kolumny
            maximum = df[column].max()

            # Konwertuj wartość na string
            maximum_str = str(maximum)

            # Pobierz tekst nagłówka kolumny z QTableView
            header_text = QtTableView.model().headerData(column, Qt.Orientation.Horizontal)

            # Wyświetl wartość w QTextBrowser
            QtTextBrowser.append(f"Maximum from column {header_text} is: {maximum_str}")
    except Exception as e:
        QtTextBrowser.append(f"Error occurred: {str(e)}")
```

Funkcja odpowiedzialna za wyznaczenie maksimum jest analogicznie skonstruowana do tej wyznaczającej minimum. Wbudowana funkcja Python max() działa bardzo podobnie do funkcji min(). Wyznacza maksymalną wartość z iterowalnej struktury danych. Wynik wypisywany jest na ekranie do wyświetlania wyników.

- Funkcja wyznaczająca medianę

```
def calculate_median(QtTextBrowser, QtTableView): #git
    try:
        for column in selected_columns:
            # Oblicz medianę z wybranej kolumny
            median = df[column].median()

            # Konwertuj wartość na string
            median_str = str(median)

            # Pobierz tekst nagłówka kolumny z QTableView
            header_text = QtTableView.model().headerData(column, Qt.Orientation.Horizontal)

            # Wyświetl wartość w QTextBrowser
            QtTextBrowser.append(f"Median from column {header_text} is: {median_str}")
    except Exception as e:
        QtTextBrowser.append(f"Error occurred: {str(e)}")
```

Mediana to środkowa wartość z tablicy liczb lub średnia arytmetyczna dwóch środkowych jeśli liczba elementów jest nieparzysta. W tym przypadku również jest to funkcja wbudowana w Pythonie. Wyznaczoną medianę wypisuję na ekranie do obliczeń.

- Funkcja wyznaczająca odchylenie standardowe

```python
def calculate_std(QtTextBrowser, QtTableView): #git
    try:
        for column in selected_columns:
            # Oblicz odchylenie standardowe z wybranej kolumny
            std = df[column].std()

            # Konwertuj wartość na string
            std_str = str(std)

            # Pobierz tekst nagłówka kolumny z QTableView
            header_text = QtTableView.model().headerData(column, Qt.Orientation.Horizontal)

            # Wyświetl wartość w QTextBrowser
            QtTextBrowser.append(f"Standard deviation from column {header_text} is: {std_str}")
    except Exception as e:
        QtTextBrowser.append(f"Error occurred: {str(e)}")
```

- Funkcja wyznaczająca średnią arytmetyczną

```python
def calculate_mean(QtTextBrowser, QtTableView): #git
    try:
        for column in selected_columns:
            # Oblicz średnią z wybranej kolumny
            mean = (df[column].mean())

            # Konwertuj wartość na string
            mean_str = str(mean)

            # Pobierz tekst nagłówka kolumny z QTableView
            header_text = QtTableView.model().headerData(column, Qt.Orientation.Horizontal)

            # Wyświetl wartość w QTextBrowser
            QtTextBrowser.append(f"Mean from column {header_text} is: {mean_str}")
    except Exception as e:
        QtTextBrowser.append(f"Error occurred: {str(e)}")
```

- Funkcja wyznaczająca korelację dla dwóch wybranych atrybutów

```python
def calculate_coorelation(comboBoxAtrybut1, comboBoxAtrybut2,PyQtTextBrowser,tableView):
    model = tableView.model()
    # pobierz indeks wybranej kolumny z comboBoxAtrybut1
    column1_index = comboBoxAtrybut1.currentIndex() + 1
    # pobierz indeks wybranej kolumny z comboBoxAtrybut2
    column2_index = comboBoxAtrybut2.currentIndex() + 1
    # pobierz dane z wybranej kolumny 1
    column1_data = [model.data(model.index(row, column1_index)) for row in range(model.rowCount())]
    # pobierz dane z wybranej kolumny 2
    column2_data = [model.data(model.index(row, column2_index)) for row in range(model.rowCount())]
    # utwórz nowy obiekt DataFrame z pobranych danych
    data = pd.DataFrame(
        {comboBoxAtrybut1.currentText(): column1_data, comboBoxAtrybut2.currentText(): column2_data})
    # data = code_data(data)
    data = data.apply(pd.to_numeric)
    correlation_matrix = data[[comboBoxAtrybut1.currentText(), comboBoxAtrybut2.currentText()]].corr()
    correlation_coefficient = correlation_matrix.iloc[0, 1]
    PyQtTextBrowser.append("Koorelacja miedzy tymi atrybutami wynosi: " + str(correlation_coefficient))
```

- Funkcja wyznaczająca zaznaczone statystyki

```
1 usage  1 Krzys3n
def calculate_checked_stats(checkBoxMin,checkBoxMax,checkBoxStd,checkBoxMdn,checkBoxMean,textBrowser,tableView):
    if (checkBoxMin.isChecked() == True):
        calculate_minimum(textBrowser,tableView)
    if (checkBoxMax.isChecked() == True):
        calculate_maximum(textBrowser,tableView)
    if (checkBoxStd.isChecked() == True):
        calculate_std(textBrowser,tableView)
    if (checkBoxMdn.isChecked() == True):
        calculate_median(textBrowser,tableView)
    if (checkBoxMean.isChecked() == True):
        calculate_mean(textBrowser,tableView)
```

6.3.3. Kodowanie generowania wykresów

- Funkcja generująca wykres zliczeń dla zaznaczonych kolumn

```
def generate_comparison_plot(tableView): #git
    model = tableView.model()
    data = pd.DataFrame()  # Utwórz pusty obiekt DataFrame

    for column_index in selected_columns:
        # Pobierz nazwę wybranej kolumny z QTableView
        column_name = model.headerData(column_index, Qt.Orientation.Horizontal)
        # Pobierz dane z wybranej kolumny
        column_data = [model.data(model.index(row, column_index)) for row in range(model.rowCount())]
        # Dodaj dane do obiektu DataFrame
        data[column_name] = column_data

    # Konwertuj dane na typ numeryczny
    data = data.apply(pd.to_numeric, errors='coerce')
    counts = data.apply(pd.Series.value_counts).fillna(0)
    counts.plot(kind='bar')
    plt.xlabel('Atrybut')
    plt.ylabel('Suma')
    # Skonstruuj tytuł wykresu na podstawie nazw zaznaczonych atrybutów
    title = "Zliczenia "

    for column_name in data.columns:
        title += str(column_name) + " + "
    title = title[:-3]  # Usuń ostatni znak "+" i spacje
    plt.title(title)
    plt.show()
```

- Funkcja generująca wykres dystrybucji dla zaznaczonych kolumn

```python
def generate_distribution_plot(tableView): #git
    model = tableView.model()
    data = pd.DataFrame()  # Utwórz pusty obiekt DataFrame

    for column_index in selected_columns:
        # Pobierz nazwę wybranej kolumny z QTableView
        column_name = model.headerData(column_index, Qt.Orientation.Horizontal)
        # Pobierz dane z wybranej kolumny
        column_data = [model.data(model.index(row, column_index)) for row in range(model.rowCount())]
        # Dodaj dane do obiektu DataFrame
        data[column_name] = column_data

    # Konwertuj dane na typ numeryczny
    data = data.apply(pd.to_numeric, errors='coerce')
    # Sumuj wartości dla każdej kolumny
    counts = data.sum()
    # Wygeneruj wykres kołowy
    plt.pie(counts.values, labels=counts.index, autopct='%1.1f%%')
    plt.axis('equal')

    # Skonstruuj tytuł wykresu na podstawie nazw zaznaczonych atrybutów
    title = "Rozkład "
    for column_name in data.columns:
        title += str(column_name) + " + "
    title = title[:-3]  # Usuń ostatni znak "+" i spacje
    plt.title(title)
    plt.show()
```

- Funkcja generująca heatmapę korelacji dla zaznaczonych kolumn

```python
def generate_correlation_heatmap(tableView): #git
    model = tableView.model()
    data = pd.DataFrame()  # Utwórz pusty obiekt DataFrame

    for column_index in selected_columns:
        # Pobierz nazwę wybranej kolumny z QTableView
        column_name = model.headerData(column_index, Qt.Orientation.Horizontal)
        # Pobierz dane z wybranej kolumny
        column_data = [model.data(model.index(row, column_index)) for row in range(model.rowCount())]
        # Dodaj dane do obiektu DataFrame
        data[column_name] = column_data

    # Konwertuj dane na typ numeryczny
    data = data.apply(pd.to_numeric, errors='coerce')

    # Oblicz macierz korelacji
    corr_matrix = data.corr()

    # Wygeneruj heatmapę korelacji
    plt.figure(figsize=(10, 8))
    sns.heatmap(corr_matrix, annot=True, cmap='coolwarm')
    plt.title('Heatmap - Korelacja')
    plt.show()
```

6.3.4. Kodowanie wczytywania oraz zapisywania plików

- Funkcja wyczytująca dane z formatu CSV

```
1 usage  Krzysztof Sendera +1 *
def load_CSV_file(comboBoxAtrybut1, comboBoxAtrybut2, comboBoxAtrybutClass, tableView):
    global df
    global df_with_labels
    global labels
    msg_box = QMessageBox()
    msg_box.setWindowTitle("Wczytywanie pliku CSV")
    msg_box.setText("Czy chcesz wczytać plik z nazwami kolumn?")
    yes_button = msg_box.addButton(QMessageBox.StandardButton.Yes)
    yes_button.setText("Tak, z nazwami")
    msg_box.addButton(QMessageBox.StandardButton.No).setText("Nie, bez nazw")
    cn_button = msg_box.addButton(QMessageBox.StandardButton.Cancel)
    cn_button.setVisible(False)
    msg_box.setDefaultButton(yes_button)
    reply = msg_box.exec()

    if reply == 16384:
        header = 0
    elif reply == 65536:
        header = None
    else:
        return

    current_dir = os.getcwd()
    filename, _ = QFileDialog.getOpenFileName(None, "Wybierz plik CSV lub DATA", current_dir,
                                              "All files (*);;Pliki CSV (*.csv);;Pliki DATA (*.data)")
    if not filename:
        # Wyjście z funkcji, jeśli nie został wybrany plik
        return
     # Wczytaj dane z pliku CSV do obiektu DataFrame z biblioteki Pandas
     df = pd.read_csv(filename, header=header)
     df_with_labels = df.copy()
     labels = df.columns.tolist()

     # Wyświetl dane za pomocą funkcji print

     model = QStandardItemModel(df.shape[0], df.shape[1])
     headers = list(df.columns)
     if reply == 16384:
         model.setHorizontalHeaderLabels(headers)

     for row in range(df.shape[0]):
         for column in range(df.shape[1]):
             item = QStandardItem(str(df.iloc[row, column]))
             model.setItem(row, column, item)

     tableView.setModel(model)

     # wczytywanie nazw kolumn do comboboxów
     column_names = df.columns.tolist()[1:]
     comboBoxAtrybut1.clear()
     comboBoxAtrybut2.clear()
     comboBoxAtrybutClass.clear()
     comboBoxAtrybut1.addItems([str(x) for x in column_names])
     comboBoxAtrybut2.addItems([str(x) for x in column_names])
     comboBoxAtrybutClass.addItems([str(x) for x in column_names])


     # Liczba kolumn minus 1
     x = len(df.columns) - 1
     # Ustawienie nazw kolumn na liczby od 0 do x
     df.columns = list(range(x + 1))

     tableView.setSelectionMode(QTableView.SelectionMode.ExtendedSelection)
     selection_model = tableView.selectionModel()
     selection_model.selectionChanged.connect(lambda:handle_selection_changed(tableView))
```

- Funkcja wczytująca dane z formatu JSON

```python
def load_JSON_file(comboBoxAtrybut1, comboBoxAtrybut2, comboBoxAtrybutClass, tableView):
    global df
    global df_with_labels
    current_dir = os.getcwd()
    filename, _ = QFileDialog.getOpenFileName(None, "Wybierz plik JSON", current_dir, "Pliki JSON (*.json)")

    if not filename:
        # Wyjście z funkcji, jeśli nie został wybrany plik
        return

    # Wczytaj dane z pliku JSON do obiektu DataFrame z biblioteki Pandas
    with open(filename, 'r') as json_file:
        data = json.load(json_file)

    df = pd.DataFrame(data)
    df_with_labels = df.copy()

    # Wyświetl dane w tabeli
    model = QStandardItemModel(df.shape[0], df.shape[1])
    headers = list(df.columns)
    model.setHorizontalHeaderLabels(headers)

    for row in range(df.shape[0]):
        for column in range(df.shape[1]):
            item = QStandardItem(str(df.iloc[row, column]))
            model.setItem(row, column, item)

    tableView.setModel(model)

    # Wczytywanie nazw kolumn do comboboxów
    column_names = df.columns.tolist()
    comboBoxAtrybut1.clear()
    comboBoxAtrybut2.clear()
    comboBoxAtrybutClass.clear()
    comboBoxAtrybut1.addItems(column_names)
    comboBoxAtrybut2.addItems(column_names)
    comboBoxAtrybutClass.addItems(column_names)

    tableView.setSelectionMode(QTableView.SelectionMode.ExtendedSelection)
    selection_model = tableView.selectionModel()
    selection_model.selectionChanged.connect(lambda: handle_selection_changed(tableView))
```

- Funkcja zapisująca dane do formatu CSV

```
1 usage  Krzysztof Sendera +1 *
def save_to_CSV():
    # Okno dialogowe z pytaniem o zapisanie z nazwami kolumn lub bez
    msg_box = QMessageBox()
    msg_box.setWindowTitle("Zapisywanie pliku CSV")
    msg_box.setText("Czy chcesz zapisać plik z nazwami kolumn?")

    yes_button = msg_box.addButton(QMessageBox.StandardButton.Yes)
    yes_button.setText("Tak")
    msg_box.addButton(QMessageBox.StandardButton.No).setText("Nie")
    cn_button = msg_box.addButton(QMessageBox.StandardButton.Cancel)
    cn_button.setVisible(False)

    msg_box.setDefaultButton(yes_button)

    reply = msg_box.exec()

    if reply == 16384:
        header = True
    elif reply == 65536:
        header = False
    else:
        return

    current_dir = os.getcwd()
    filename, _ = QFileDialog.getSaveFileName(None, "Zapisz plik CSV", current_dir, "Pliki CSV (*.csv)")
    if filename:
        df_with_labels.to_csv(filename, index=False, header=header)
```

- Funkcja zapisująca dane do formatu JSON

```
1 usage  Krzys3n
def save_to_JSON():
    current_dir = os.getcwd()
    filename, _ = QFileDialog.getSaveFileName(None, "Zapisz plik JSON", current_dir, "Pliki JSON (*.json)")

    if filename:
        # Zakładam, że masz DataFrame o nazwie df_with_labels, które chcesz zapisać do pliku JSON.
        # Jeśli używasz innej zmiennej, zmień ją w odpowiednim miejscu.
        data_to_save = df.to_dict(orient='records')

        with open(filename, 'w') as json_file:
            json.dump(data_to_save, json_file, indent=4)
```

6.3.5. Kodowanie generowania raportu PDF

```python
def generate_pdf(QtTextBrowser):
    # Wyświetl okno dialogowe do wyboru miejsca zapisu pliku
    file_dialog = QFileDialog()
    file_dialog.setWindowTitle("Save PDF")
    file_dialog.setFileMode(QFileDialog.FileMode.AnyFile)
    file_dialog.setAcceptMode(QFileDialog.AcceptMode.AcceptSave)
    file_dialog.setNameFilter("PDF Files (*.pdf)")
    file_dialog.setDefaultSuffix("pdf")

    if file_dialog.exec() == QFileDialog.DialogCode.Accepted:
        file_path = file_dialog.selectedFiles()[0]
    else:
        return  # Użytkownik anulował wybór pliku

    if file_path:
        # Pobierz tekst z QTextBrowser
        text = QtTextBrowser.toPlainText()
        # Utwórz nowy obiekt Canvas
        c = canvas.Canvas(file_path)
        # Ustaw czcionkę i rozmiar tekstu
        c.setFont("Helvetica", 12)
        # Wypisz tekst w pliku PDF
        lines = text.split("\n")
        y = 800  # Wysokość początkowa
        for line in lines:
            c.drawString(50, y, line)
            y -= 20  # Zmniejsz wysokość dla kolejnej linii
        # Zamknij plik PDF
        c.save()
```

6.3.6. Kodowanie wyświetlania informacji o przyciskach po najechaniu na nie kursorem myszy

```python
## Sekcja wyświetlania informacji:
# wyświetlanie informacji o przyciskach:
1 usage  Krzys3n *
def switch_dictionary_buttons(button_name): #git
    switcher = {
        "pushButtonMinimum": "Calculates the minimum of selected columns data",
        "pushButtonMaximum": "Calculates the maximum of selected columns data",
        "pushButtonStd": "Calculates the standard deviation of selected columns data",
        "pushButtonMedian": "Calculates the median of selected columns data",
        "pushButtonMean": "Calculates the arithmetic average of selected columns data",
        "pushButtonClear": "Clears the result window",
        "pushButtonCalcChecked": "Designates selected descriptive statistics",
        "pushButtonKoorelacja": "Calculates the correlation of selected attributes and generates a correlation heatmap",
        "pushButtonPorownaj": "Generates a comparison plot of selected columns data",
        "pushButtonDystrybucja": "Generates a distribution plot of selected columns data",
        "pushButtonHeatmap": "Generates a coorelation Heatmap of selected columns data",
        "pushButtonClass": "Provides classification of selected attribute for selected data",
        "pushButtonCluster": "Provides clusterization  for selected clusters for selected data"

    }
    return switcher.get(button_name, "No information available for the button")


12 usages  Krzys3n
def on_button_enter(event: QEnterEvent, button_name: str, QtTextBrowser): #git
    info = switch_dictionary_buttons(button_name)
    QtTextBrowser.setText(info)
    QtTextBrowser.setAlignment(Qt.AlignmentFlag.AlignCenter)
```

6.3.7. Kodowanie stylów, które umożliwiają wybór motywu aplikacji

- Funkcja zmieniająca motyw na jasny

```
1 usage  Krzys3n *
def change_to_lightmode(window,table_view):
    window.setStyleSheet("")
    table_view.horizontalHeader().setVisible(True)
    table_view.verticalHeader().setVisible(True)
    table_view.setStyleSheet("""
            background-color: #fffff;
            color: black;
            font-family: Titillium;
            font-size: 18px;

        }
            """)

    table_view.horizontalHeader().setStyleSheet("QHeaderView::section { background-color: #fffff; color: black; }")
    table_view.verticalHeader().setStyleSheet("QHeaderView::section { background-color: #fffff; color: black; }")
```

- Funkcja zmieniająca motyw na ciemny

```
## Sekcja styli
1 usage  Krzys3n
def change_to_darkmode(window,table_view,textBrowserInfo):
    window.setStyleSheet("""
            background-color: #262626;
            color: white;
            font-family: Titillium;
            font-size: 14px;

        }
            """)
    table_view.setStyleSheet("""
            background-color: #262626;
            color: white;
            font-family: Titillium;
            font-size: 18px;

        }
            """)
    table_view.horizontalHeader().setStyleSheet("QHeaderView::section { background-color: #262626; color: white; }")
    table_view.verticalHeader().setStyleSheet("QHeaderView::section { background-color: #262626; color: white; }")
```

6.3.8.

6.4. Podłączenie projektu do bazy danych z użytkownikami

6.5.

6.6. Eksportowanie aplikacji użytkowej.

Aplikację użytkową wyeksportujemy za pomocą oprogramowania pyinstaller. Jest to narzędzie służące do pakowania aplikacji napisanych w języku Python w jednym pliku wykonywalnym. Jest to narzędzie umożliwiające łatwe tworzenie samodzielnych plików wykonywalnych (.exe na systemach Windows, .app na systemach macOS, itp.) z kodu źródłowego Pythona.

Główne zalety PyInstallera to:

- PyInstaller pozwala na utworzenie jednobazowych plików wykonywalnych, co oznacza, że cała aplikacja jest zawarta w jednym pliku, co ułatwia przenoszenie i udostępnianie aplikacji.
- Obsługa wielu platform: PyInstaller jest wieloplatformowy, co oznacza, że możesz używać go do pakowania aplikacji na różne systemy operacyjne, takie jak Windows, macOS i Linux.
- Obsługa zależności: PyInstaller jest w stanie automatycznie wykrywać i dołączać zależności Twojej aplikacji, takie jak moduły biblioteki standardowej Pythona oraz zewnętrzne biblioteki.
- Prosta obsługa: Proces używania PyInstallera jest stosunkowo prosty. Wystarczy uruchomić kilka poleceń w terminalu lub wierszu polecenia, aby uzyskać gotowy plik wykonywalny.

  Moją aplikację wyeksportuję do jednego pliku za pomocą polecenia
  `pyinstaller -F MainWindow.py`
  MainWindow.py to główny skrypt mojego projektu w którym zawarty jest kod uruchamiający całą aplikację.
  Atrybut -F oznacza, że projekt ma zostać spakowany do jednego pliku. Można również spakować do do folderu wraz z różnymi skryptami i bibliotekami używając tego polecenia bez atrybutu -F.

6.7. Kodowanie funkcji odpowiedzialnych za edycje, aktualizację oraz wypisywanie danych.

7. Sposób użytkowania aplikacji
8. Testowanie Błędów.
9. Podsumowanie
10. Bibliografia

10.1. pandas documentation — pandas 2.1.1 documentation (pydata.org)

10.2. Matplotlib documentation — Matplotlib 3.8.0 documentation

10.3. seaborn: statistical data visualization — seaborn 0.13.0 documentation (pydata.org)

10.4. Qt for Python

10.5. Encyklopedia Zarządzania (mfiles.pl)
10.6. PyQt6 · PyPI
10.7. PyQt6 Tutorial 2023, Create Python GUIs with Qt
10.8. scikit-learn: machine learning in Python — scikit-learn 1.3.2 documentation
10.9. scikit-learn · PyPI
10.10. seaborn: statistical data visualization — seaborn 0.13.0 documentation (pydata.org)
10.11. Python Tutorials – Real Python