**Model Development Phase Template**

|  |  |
| --- | --- |
| Date | 15 July 2024 |
| Team ID | 739849 |
| Project Title | Doctors Annual Salary Prediction |
| Maximum Marks | 10 Marks |

**Initial Model Training Code, Model Validation and Evaluation Report**

The initial model training code will be showcased in the future through a screenshot. The model validation and evaluation report will include a summary and training and validation performance metrics for multiple models, presented through respective screenshots.

**Initial Model Training Code (5 marks):**

For the project titled "Doctor's Salary Prediction Using ML," the initial model training code is a crucial part of the development process. This code sets up the machine learning environment, prepares the data, and trains the model to predict doctors' salaries based on various features.

from pathlib import PureWindowsPath

import numpy as np

import pandas as pd

import warnings

warnings.filterwarnings("ignore")

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.preprocessing import LabelEncoder

from sklearn.model\_selection import train\_test\_split,GridSearchCV

df = pd.read\_excel(r'C:\Users\akhil\OneDrive\Desktop\soumyasri project\Dataset\NewDoctorsPay.xlsx')

df

df.info()

df.isnull().sum()

le = LabelEncoder()

df['Specialty'] = le.fit\_transform(df['Specialty'])

#df['Annual Income'] = le.fit\_transform(df['Annual Income'])

df['Feel Fairly Compensated'] = le.fit\_transform(df['Feel Fairly Compensated'])

df['Overall Career Satisfaction'] = le.fit\_transform(df['Overall Career Satisfaction'])

df['Satisfied Income'] = le.fit\_transform(df['Satisfied Income'])

df['Would Choose Medicine Again'] = le.fit\_transform(df['Would Choose Medicine Again'])

df['Would Choose the Same Specialty'] = le.fit\_transform(df['Would Choose the Same Specialty'])

df['Survey Respondents by Specialty'] = le.fit\_transform(df['Survey Respondents by Specialty'])

df.describe()

df.drop(columns=["% Female","Difference in Earnings between Physicians who Feels Fairly vs Unfairly Paid"],axis=1,inplace=True)

df.describe()

sns.countplot(df['Annual Income'])

plt.show()

sns.boxplot(x='Annual Income',y='Overall Career Satisfaction',data=df)

plt.show()

sns.pairplot(df)

plt.show()

x = df.drop(['Annual Income'],axis=1)

y = df['Annual Income']

from sklearn.model\_selection import train\_test\_split,GridSearchCV

x\_train,x\_test,y\_train,y\_test = train\_test\_split(x,y,test\_size = 0.3,random\_state = 42)

from sklearn.linear\_model import LinearRegression

reg = LinearRegression()

reg.fit(x\_train,y\_train)

from sklearn.metrics import r2\_score

from sklearn.metrics import mean\_squared\_error

y\_train\_pred = reg.predict(x\_train)

y\_test\_pred = reg.predict(x\_test)

y\_train\_pred[:5]

y\_test\_pred[:5]

r2\_score(y\_train,y\_train\_pred) \* 100

mean\_squared\_error(y\_train,y\_train\_pred

r2\_score(y\_test,y\_test\_pred)\*100

mean\_squared\_error(y\_test,y\_test\_pred)

from sklearn.ensemble import RandomForestRegressor

rf=RandomForestRegressor(n\_estimators=100,random\_state=42)

rf.fit(x\_train,y\_train)

y\_train\_pred=rf.predict(x\_train)

y\_test\_pred=rf.predict(x\_test)

r2\_score(y\_train,y\_train\_pred)

mean\_squared\_error(y\_train,y\_train\_pred)

r2\_score(y\_test,y\_test\_pred)

mean\_squared\_error(y\_test,y\_test\_pred)

from sklearn.tree import DecisionTreeRegressor

dtr=DecisionTreeRegressor(random\_state=42)

dtr.fit(x\_train,y\_train)

y\_train\_pred=dtr.predict(x\_train)

y\_train\_pred=dtr.predict(x\_test)

y\_train\_pred[:5]

y\_train\_pred[:5]

r2\_score(y\_train,y\_train\_pred)\*100

r2\_score(y\_train,y\_train\_pred)\*100

r2\_score(y\_train,y\_train\_pred)\*100

mean\_squared\_error(y\_test,y\_test\_pred)

import xgboost as xgb

xg\_reg=xgb.XGBRegressor()

xg\_reg.fit(x\_train,y\_train)

xg\_reg.fit(x\_train,y\_train)

y\_train\_pred=xg\_reg.predict(x\_train)

y\_test\_pred=xg\_reg.predict(x\_test)

r2\_score(y\_train,y\_train\_pred)\*100

mean\_squared\_error(y\_train,y\_train\_pred)

r2\_score(y\_test,y\_test\_pred)\*100

mean\_squared\_error(y\_test,y\_test\_pred)

r2\_score(y\_train,y\_train\_pred)\*100

reg.predict([[11,5,1,5,18,0,1]])

reg.predict([[23,9,6,9,1,12,4]])

reg.predict([[10,7,9,7,16,9,0]])

**Model Validation and Evaluation Report (5 marks):**

|  |  |  |
| --- | --- | --- |
| **Model** | **Summary** | **Training and Validation Performance Metrics** |
| Model 1 | **Linear Regression**: Simple linear model that assumes a linear relationship between the features and the target variable. | **Training MSE**: 1405231186.3519106  **Validation MSE**:3715045452.16928  **R-squared**: 27.269167796800964 |
| Model 2 | **Random Forest regression:**Random Forest Regression is a supervised machine learning algorithm that uses an ensemble of decision trees to predict continuous target variables | |  | | --- | | **Training MSE**: 458713655.5555556  **Validation MSE**: 3631440587.5  **R-squared**: 0.289059314547212 | |
| Model 3 | **Decision tree regreesor :**A Decision Tree Regressor is a supervised machine learning algorithm used for predicting continuous target variables. | **Training MSE**: 0.0  **Validation MSE**:3565125000.0  **R-squared**: 100.0 |
| Model 4 | **XGBRegressor :** The XGBRegressor is a powerful machine learning algorithm from the XGBoost library, designed for regression tasks. | **Training MSE**:0.000132921006944444445  **Validation MSE**:3566414555.666992  **R-squared**:99.999999999 |