Executive Summary

ML Model Development Lifecycle and Project Outcomes

The methodical lifecycle of creating machine learning (ML) models is intended to yield precise, trustworthy, and morally sound results. The goal of this study was to develop a predictive machine learning model using the Titanic dataset in order to categorize whether or not passengers survived by taking into account a number of characteristics, including age, sex, class, and family size. Data preparation, exploratory analysis, feature engineering, model selection, training, assessment, and deployment were all included in the workflow.

Data Collection and Preparation

Gathering the Titanic dataset from CSV files and cleaning it up by deleting unnecessary columns like Name and PassengerId and filling in missing variables (such as imputing Age, Cabin, and Embarked) were the first steps.

```
[ ] from google.colab import drive
    drive.mount('/content/drive')

    Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content,
```

```
[ ] # Let's load the dataset and inspect the first few rows to understand its structure.
    import pandas as pd
    # Load the dataset
    df_train = pd.read_csv("/content/drive/MyDrive/ANIL_ASSIGN/Titanic/train.csv")
    df_test = pd.read_csv("/content/drive/MyDrive/ANIL_ASSIGN/Titanic/test.csv")
    target = pd.read_csv("/content/drive/MyDrive/ANIL_ASSIGN/Titanic/gender_submission.csv")
```

Exploratory Data Analysis (EDA)

The associations between different variables and survival rates were examined using exploratory data analysis (EDA). The following important parameters were highlighted by visual aids such histograms, box plots, and correlation matrices: age, family size, Pclass, and sex. First-class passengers, women, and children were shown to have better survival rates. Decisions around feature engineering and model selection were informed by these findings.

```
[ ] # Display the last 5 rows of the df_train dataset
    df_train.tail()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.00	NaN	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.00	B42	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.45	NaN	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.00	C148	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.75	NaN	Q

```
[ ] # Calculate the total number of male passengers who survived by filtering rows where 'Sex' is 'mal
    df_train[df_train['Sex']=='male']['Survived'].sum()

109
```

```
[ ] # Calculate the total number of female passengers who survived by filtering rows where 'Sex' is 'f
    df_train[df_train['Sex']=='female']['Survived'].sum()

233
```

Feature Engineering and Transformation

By creating predictive features, feature engineering improved the dataset. SibSp and Parch were combined to create FamilySize, while travelers without family were identified by IsAlone. Numerical features were standardized for homogeneity, while categorical data like Sex and Embarked were translated using one-hot encoding. Before modeling, column transformers made it easier to preprocess both categorical and numerical data.

```
[ ] # Create the FamilySize column
    df_train['FamilySize'] = df_train['SibSp'] + df_train['Parch']

    # Create the IsAlone column
    df_train['IsAlone'] = (df_train['FamilySize'] == 0).astype(int)
```

Model Selection and Training

This binary classification assignment evaluated a number of machine learning models, including K-Nearest Neighbors (KNN), Decision Tree Classifier (DTC), Support Vector Classifier (SVC), and Logistic Regression (LR). A training-validation split was used to train the models in order to prevent overfitting and encourage efficient generalization. In order to guarantee consistent operations, pipelines were used to streamline preprocessing and modeling.

```
[ ]  from sklearn.neighbors import KNeighborsClassifier
     from sklearn.tree import DecisionTreeClassifier
     from sklearn.svm import SVC
     from sklearn.linear_model import LogisticRegression


     model_KNN = make_pipeline(preprocessor, KNeighborsClassifier())
     model_DTC = make_pipeline(preprocessor, DecisionTreeClassifier())
     model_SVC = make_pipeline(preprocessor, SVC())
     model_LR = make_pipeline(preprocessor, LogisticRegression())
```

Model Evaluation

Using a test set, models were assessed for accuracy, precision, recall, and F1 score.

- K-Neighbors Classifier: 0.8445
- Decision Tree Classifier: 0.8780
- Support Vector Classifier: 0.9522
- Logistic Regression: 0.9474

While Logistic Regression worked well for modeling linear correlations, the SVC was notable for its capacity to identify intricate patterns. Although it performed well in handling categorical data, the Decision Tree Classifier displayed indications of possible overfitting. The K-Neighbors Classifier, which relies on proximity-based patterns, fared moderately despite being simpler. For situations where accuracy is a top concern, both SVC and logistic regression were very well-suited.

```
[ ] pd.DataFrame(zip([model_KNN.score(X_test, Y_test)],
                     [model_DTC.score(X_test, Y_test)],
                     [model_SVC.score(X_test, Y_test)],
                     [model_LR.score(X_test, Y_test)]
                    ),
                 columns=['K-Neighbors Classifier',
                          'Decision Tree Classifier',
                          'Support Vector Classifier',
                          'Logistic Regression']
                )
```

	K-Neighbors Classifier	Decision Tree Classifier	Support Vector Classifier	Logistic Regression
0	0.844498	0.87799	0.952153	0.947368

```
Classification Report for Support Vector Classifier:

              precision    recall  f1-score   support

Not Survived       0.97      0.95      0.96       266
    Survived       0.92      0.95      0.94       152

    accuracy                           0.95       418
   macro avg       0.95      0.95      0.95       418
weighted avg       0.95      0.95      0.95       418
```

```
Classification Report for Logistic Regression:

              precision    recall  f1-score   support

Not Survived       0.97      0.95      0.96       266
    Survived       0.91      0.95      0.93       152

    accuracy                           0.95       418
   macro avg       0.94      0.95      0.94       418
weighted avg       0.95      0.95      0.95       418
```

```
Classification Report for Decision Tree Classifier:

              precision    recall  f1-score   support

Not Survived       0.89      0.91      0.90       266
    Survived       0.84      0.81      0.82       152

    accuracy                           0.87       418
   macro avg       0.86      0.86      0.86       418
weighted avg       0.87      0.87      0.87       418
```

```
Classification Report for K-Neighbors Classifier:

              precision    recall  f1-score   support

Not Survived       0.88      0.88      0.88       266
    Survived       0.78      0.79      0.79       152

    accuracy                           0.84       418
   macro avg       0.83      0.83      0.83       418
weighted avg       0.84      0.84      0.84       418
```

Model Deployment and Outcomes

To ensure that it does not reflect biases connected to social and demographic aspects like gender and class, the final model was chosen based on its accuracy and fairness. It has been used for assessment and instruction, highlighting the elements that affected Titanic survival.

Conclusion

The ML model development lifecycle, from data collection and preparation to model deployment, was crucial in building a robust predictive system. The project not only demonstrated the practical steps of building an ML model but also highlighted ethical considerations and the importance of model fairness in decision-making. By carefully evaluating and selecting the best-performing model, the project provides valuable insights into the complexities of predictive modeling and its real-world implications.