Survival Prediction on the Titanic Dataset

Prepared by: Vishnu N Date: 21/11/2024

Affiliation: Data Science Enthusiast at Government College of Technology

LinkedIn: https://www.linkedin.com/in/vishnu-n-

121108293?utm_source=share&utm_campaign=share_via&utm_content=profile&utm_medium=a

ndroid_app

Abstract

The Titanic disaster of 1912 remains one of the most infamous maritime tragedies. This project aims to predict passenger survival using machine learning models trained on the Titanic dataset. The dataset was pre-processed to handle missing values, normalize numerical features, and encode categorical variables. Various models, including Logistic Regression, Random Forest, and AdaBoost, were employed and optimized using GridSearchCV. The Random Forest model achieved the highest accuracy of 83.4%, followed by AdaBoost at 85.6% with hyperparameter tuning. This project demonstrates the power of data-driven insights in understanding historical events and predicting outcomes.

Introduction

The sinking of the Titanic is a historical tragedy that has fascinated researchers and the public for decades. Among the 2,224 passengers and crew, only 710 survived, raising questions about the factors influencing survival. Using the Titanic dataset, this project aims to predict survival probabilities based on passenger demographics, ticket class, and other features. By applying machine learning techniques, we explore the relationship between these factors and survival, identify key predictors, and develop a model to provide reliable predictions.

The dataset, sourced from Kaggle, consists of 891 records with features such as passenger age, gender, class, and fare. The primary objective is to preprocess the data, apply machine learning models, and optimize their performance through hyperparameter tuning.

Exploratory Data Analysis (EDA)

The dataset was analyzed to identify key patterns and relationships between variables:

1. Gender and Survival:

- o Females had a significantly higher survival rate than males.
- o **Visualization**: A count plot showed females were more likely to survive.

2. Passenger Class and Survival:

 Passengers in first class had a higher survival rate compared to those in lower classes. • Visualization: A bar plot depicted survival rates by class.

3. Age Distribution:

- The average passenger age was 29.7 years.
- o Children and younger passengers had higher survival probabilities.

4. Missing Values:

- o 177 missing values in the Age column were filled with the mean.
- o 2 missing values in Embarked were filled with the mode.

Key graphs were plotted using Seaborn and Matplotlib to visualize survival distributions across features like age, gender, and class.

Methodology

The project followed a structured approach:

1. Data Preprocessing:

- o Removed irrelevant columns (Passengerld, Name, Ticket, Cabin).
- Normalized Age and Fare using MinMaxScaler and StandardScaler.
- o Encoded categorical variables (Sex, Embarked) using one-hot encoding.

2. Feature Engineering:

- o Ensured all features were numerical for compatibility with ML models.
- o Created dummy variables for categorical features.

3. Model Selection and Training:

- o **Models Used**: Logistic Regression, Random Forest, AdaBoost.
- o **Training**: Split the dataset into 75% training and 25% testing sets.

4. Hyperparameter Tuning:

- o Used GridSearchCV to find the best parameters for each model.
- Scoring was based on accuracy and other relevant metrics.

5. **Prediction Function**:

 Implemented a function to predict survival based on user-input features, making the model interactive.

Results

The performance of different models was evaluated based on accuracy:

Tabular Representation of best Parameters:

Model	Best Parameters (via GridSearchCV)	Accuracy on Test Set
Logistic Regression	{'C': 1.0, 'solver': 'lbfgs', 'penalty': 'l2'}	82.0%
Random Forest	{'criterion': 'entropy', 'max_depth': 10, 'n_estimators': 50}	83.4%
AdaBoost	{'algorithm': 'SAMME.R', 'learning_rate': 0.01, 'n_estimators': 200}	85.6%

Insights:

- Ensemble methods like Random Forest and AdaBoost outperformed Logistic Regression.
- Hyperparameter tuning significantly improved model performance, especially for AdaBoost.

Discussion

Key Observations:

1. Important Features:

- o Gender, passenger class, and fare were the most influential factors for survival.
- Females and first-class passengers had higher survival rates.

2. Model Performance:

- o AdaBoost achieved the highest accuracy due to its iterative boosting mechanism.
- o Random Forest performed well but slightly underperformed AdaBoost.

3. Challenges:

- Limited dataset size restricted model training.
- Missing values required careful imputation to avoid data loss or bias.

Limitations:

• The dataset lacks additional features like health conditions or travel companions, which might further influence survival.

Future Work:

- Experiment with more advanced models (e.g., XGBoost, Gradient Boosting).
- Use feature selection techniques to identify the most impactful features.

Conclusion

This project successfully predicted Titanic passenger survival using machine learning techniques. By preprocessing the data and applying models like Logistic Regression, Random Forest, and AdaBoost, we achieved an accuracy of up to 85.6%. The results highlight the significance of gender, class, and fare in survival predictions.

Future iterations could incorporate larger datasets, additional features, or advanced models for further improvement. This project demonstrates the value of machine learning in understanding historical events and deriving actionable insights.

original_data = data data.head() PassengerId Survived Pclass	train_test_split,GridSearchCV AdaBoostClassifier _absolute_error,confusion_matrix,accuracy_score,precision_score,fl_score
	Name Sex Age SibSp Parch Ticket Fare Cabin Embarked nd, Mr. Owen Harris male 22.0 1 0 A/5 21171 7.2500 NaN S Florence Briggs Th female 38.0 1 0 PC 17599 71.2833 C85 C
3 4 1 1 Futrelle, Mrs. Jacques Ho 4 5 0 3 Aller	eikkinen, Miss. Laina female 26.0 0 0 STON/O2. 3101282 7.9250 NaN S leath (Lily May Peel) female 35.0 1 0 113803 53.1000 C123 S In, Mr. William Henry male 35.0 0 0 373450 8.0500 NaN S
Feature Engineering Let us drop the unwanted features in the dataset data = data.drop(columns = ["PassengerId", "Name", "Tidata.head()	.cket", "Cabin"], axis = 1)
Survived Pclass Sex Age SibSp Parch Fare Emb 0 0 3 male 22.0 1 0 7.2500 1 1 1 female 38.0 1 0 71.2833 2 1 3 female 26.0 0 0 7.9250	s C S
3 1 1 female 35.0 1 0 53.1000 4 0 3 male 35.0 0 0 8.0500 Now the Unwanted Data Features have been Droped Now we should Check whether the Dataset has any empty Value	s s
data.isnull().sum() Survived 0 Pclass 0 Sex 0 Age 177	
	ture and the 2 values are missing in the EmbarkedLet us fix that
<pre>data["Age"] = data["Age"].fillna(data["Age"].mean()) data["Embarked"] = data["Embarked"].fillna(data["Emb data.isnull().sum()) Survived 0 Pclass 0 Sex 0 Age 0 SibSp 0</pre>	
Parch 0 Fare 0 Embarked 0 dtype: int64 Now it is clear that there is no missing values in any of the Feato Let us display the overview of our Dataset	ure in our Dataset
count 891.000000 891.000000 891.000000 891.000000 891.0	Parch Fare 000000 891.000000 381594 32.204208
std 0.486592 0.836071 13.002015 1.102743 0.8 min 0.000000 1.000000 0.420000 0.000000 0.0	806057 49.693429 000000 0.000000 000000 7.910400
	000000 31.000000 000000 512.329200
<pre>scaler = MinMaxScaler() data["Age"] = scaler.fit_transform(data[["Age"]]) scaler = StandardScaler() data["Fare"] = scaler.fit_transform(data[["Fare"]]) data.head()</pre> Survived Pclass Sex Age SibSp Parch Fare	e Embarked
0 0 3 male 0.271174 1 0 -0.502445 1 1 1 female 0.472229 1 0 0.786845 2 1 3 female 0.321438 0 0 -0.488854 3 1 1 female 0.434531 1 0 0.420736	5 C 4 S
4 0 3 male 0.434531 0 0 -0.486333 data = pd.get_dummies(data,columns = ["Sex","Embarke data.head() Survived Pclass Age SibSp Parch Fare Sex_r	
1 1 1 0.472229 1 0 0.786845 F 2 1 3 0.321438 0 0 -0.488854 F 3 1 1 0.434531 1 0 0.420730 F	True False True False False False False False True False False True False False True
Lets us confirm that our Dataset Only contains the Numerical va data.info() <class 'pandas.core.frame.dataframe'=""> RangeIndex: 891 entries, 0 to 890</class>	True False True
Data columns (total 9 columns): # Column Non-Null Count Dtype	
5 Fare 891 non-null float64 6 Sex_male 891 non-null bool 7 Embarked_Q 891 non-null bool 8 Embarked_S 891 non-null bool dtypes: bool(3), float64(2), int64(4) memory usage: 44.5 KB Let us Visulaize the relationsip between the Dependent and the	Independent variables of our Dataset
sns.countplot(data=data,x = "Pclass",hue = "Survived plt.show()	Survived 0 1
300 - 250 - tin 200 -	
150 -	
Pclass sns.catplot (data=original_data,x = "Sex",hue = "Sur" <seaborn.axisgrid.facetgrid 0x1e32c32e480="" at=""></seaborn.axisgrid.facetgrid>	<pre>rvived",y ='Age',palette = 'Set1',kind = 'bar')</pre>
30 -	
20 - 80 15 -	Survived 0 1
5 -	
<pre>male</pre>	
<pre>("Random Forest", RandomForestClassifier), ("Ada Boost", AdaBoostClassifier)] Let us know about the Parameters of the Linear and the Logistic def display_params(estimator):</pre>	> Regressors.
<pre>signature = inspect.signature(estimator) print(estimator) for params in signature.parameters.values(): print(f"{params.name}:{params.default}") print("-"*40) for name, model in linear_models: display_params(model) <class 'sklearn.linear_modelbase.linearregression'=""> fit_intercept:True</class></pre>	
fit_intercept:True copy_X:True n_jobs:None positive:False	sion'>
<pre>C:1.0 fit_intercept:True intercept_scaling:1 class_weight:None random_state:None solver:lbfgs max_iter:100 multi_class:deprecated</pre>	
<pre>verbose:0 warm_start:False n_jobs:None ll_ratio:None</pre>	
<pre><class 'sklearn.ensembleforest.randomforestclassifi="" criterion:gini="" max_depth:none="" max_features:sqrt="" max_leaf_nodes:none<="" min_samples_leaf:1="" min_samples_split:2="" min_weight_fraction_leaf:0.0="" n_estimators:100="" pre=""></class></pre>	er'>
<pre>min_impurity_decrease:0.0 bootstrap:True oob_score:False n_jobs:None random_state:None verbose:0 warm_start:False class_weight:None</pre>	
<pre>ccp_alpha:0.0 max_samples:None monotonic_cst:None</pre>	.ssifier'>
Now we got the parameters and let us find the best Parameters Hyper Parameter Tuning	That gives the best Optimal Soluctions
<pre>Hyperparameter Tuning for Linear Regressio x=data.drop("Survived",axis = 1) y = data["Survived"] parameters = { 'fit_intercept':[True,False], 'copy_X':[True,False],</pre>	n Model
<pre>'n_jobs': [None] + list(range(1, 12)), 'positive': [False, True] } x_train, x_test, y_train, y_test = train_test_split(x, y) lin_model = LinearRegression() grid_search = GridSearchCV(lin_model, parameters, cv = grid_search.fit(x_train, y_train) best_parameters = grid_search.best_params_ best_score = grid_search.best_score_</pre>	
<pre>print(f"The Best Parameters: {best_parameters}") print(f"Best Accuracy: {best_score}") The Best Parameters: {'copy_X': True, 'fit_intercept' Best Accuracy: nan C:\Users\vinu0\anaconda3\Lib\site-packages\sklearn\mo nan nan nan nan nan nan nan nan nan nan</pre>	odel_selection_search.py:1102: UserWarning: One or more of the test scores are non-finite: [nan nan nan nan nan nan nan nan nan na
nan	
<pre>fare_min = data["Fare"].min() fare_max = data["Fare"].max() fare_scaler = MinMaxScaler(feature_range=(0, 1)) fare_scaler.fit(np.array([fare_min, fare_max]).reshadef check():</pre>	
	the Passenger that They have Booked: ")) : ")) the Count of the passenger's Sibling and Spouse that they have also travelled with them: ")) the Count of the passenger's Parent and Children that they have also travelled with them: ")) to of the Passenger that they have travelled: ")) [0][0]
<pre>Embarked_from_Queenstown = int(input("Whether th Embarked_from_Southampton = int(input("Whether t</pre>	ne Passenger Embarked from Queenstown (1 for Yes, 0 for No): ")) The Passenger Embarked from Southampton (1 for Yes, 0 for No): ")) The Passenger Embarked from Southampton (1 for Yes, 0 for No): ")) The Passenger Embarked from Southampton (1 for Yes, 0 for No): ")) The Passenger Embarked from Queenstown (1 for Yes, 0 for No): ")) The Passenger Embarked from Queenstown (1 for Yes, 0 for No): ")) The Passenger Embarked from Queenstown (1 for Yes, 0 for No): ")) The Passenger Embarked from Queenstown (1 for Yes, 0 for No): ")) The Passenger Embarked from Queenstown (1 for Yes, 0 for No): ")) The Passenger Embarked from Queenstown (1 for Yes, 0 for No): ")) The Passenger Embarked from Queenstown (1 for Yes, 0 for No): ")) The Passenger Embarked from Queenstown (2 for No): ")) The Passenger Embarked from Queenstown (2 for No): ")) The Passenger Embarked from Queenstown (2 for No): ")) The Passenger Embarked from Queenstown (2 for No): ")) The Passenger Embarked from Queenstown (2 for No): ")) The Passenger Embarked from Queenstown (2 for No): ")) The Passenger Embarked from Queenstown (2 for No): ")) The Passenger Embarked from Queenstown (2 for No): ")) The Passenger Embarked from Queenstown (2 for No): ")) The Passenger Embarked from Queenstown (2 for No): ") The Passenger Embarked from Queenstown (2 for No): ") The Passenger Embarked from Queenstown (2 for No): ") The Passenger Embarked from Queenstown (2 for No): ") The Passenger Embarked from Queenstown (2 for No): ") The Passenger Embarked from Queenstown (2 for No): ") The Passenger Embarked from Queenstown (2 for No): ") The Passenger Embarked from Queenstown (2 for No): ") The Passenger Embarked from Queenstown (2 for No): ") The Passenger Embarked from Queenstown (2 for No): ") The Passenger Embarked from Queenstown (2 for No): ") The Passenger Embarked from Queenstown (2 for No): ") The Passenger Embarked from Queenstown (2 for No): ") The Passenger Embarked from Queenstown (2 for No): ") The Passenger Embarked from Queenstown (2 for No): ") The Passenger
<pre>f"count_of_parents_and_children = {count_of_pa f"Fare = {fare_scaler.inverse_transform(np.arr f"Male = {Male} \n" f"Embarked_from_Queenstown = {Embarked_from_Qu f"Embarked_from_Southampton = {Embarked_from_S if prediction == 1:</pre>	arents_and_children} \n" ray([[Fare]]))[0][0]} \n" neenstown} \n"
else:	nger has Unfortunately not Survived in the Titanic Disaster.")
<pre>model.fit(x_train, y_train) y_pred_lin = model.predict(x_test) from sklearn.metrics import mean_squared_error mse = mean_squared_error(y_test, y_pred_lin) print(f"Mean Squared Error: {mse}") check()</pre> Mean Squared Error: 0.12967317999626324	
Enter input values for the given features: [-28.02855223] Passenger Class = 1 Age = 67 Count_of_siblings_and_spouse = 1 count_of_parents_and_children = 2 Fare = 146.0 Male = 1	
<pre>Embarked_from_Queenstown = 1 Embarked_from_Southampton = 0 Based on the values given, the Passenger has Unfortun C:\Users\vinu0\anaconda3\Lib\site-packages\sklearn\ba warnings.warn(Now let us check for the Other Linear Model Logistic Regression</pre>	nse.py:493: UserWarning: X does not have valid feature names, but LinearRegression was fitted with feature names
<pre>parameters = { 'penalty': ['12'], 'tol': [1e-4, 1e-3], 'C': [0.01, 0.1, 1.0], 'fit_intercept': [True, False], 'solver': ['lbfgs', 'sag', 'saga'],</pre>	compatible with the others so we have two dictionaries I1 and I2 to avoid the errors_
'max_iter': [100, 200],	
<pre>'multi_class': ['auto'], 'n_jobs': [-1] } parameters_liblinear = { 'penalty': ['l1', 'l2'], 'dual': [False], 'tol': [1e-4, 1e-3], 'C': [0.01, 0.1, 1.0],</pre>	
<pre>'multi_class': ['auto'], 'n_jobs': [-1] } parameters_liblinear = { 'penalty': ['l1', 'l2'], 'dual': [False], 'tol': [le-4, le-3], 'C': [0.01, 0.1, 1.0], 'fit_intercept': [True, False], 'solver': ['liblinear'], 'max_iter': [100, 200], 'multi_class': ['auto'], 'n_jobs': [-1] } model = LogisticRegression() grid_search = GridSearchCV(model, [parameters, param</pre>	meters_liblinear], cv=5, scoring='accuracy', n_jobs=-1)
<pre>'multi_class': ['auto'], 'n_jobs': [-1] } parameters_liblinear = { 'penalty': ['ll', 'l2'], 'dual': [False], 'tol': [le-4, le-3], 'C': [0.01, 0.1, 1.0], 'fit_intercept': [True, False], 'solver': ['liblinear'], 'max_iter': [100, 200], 'multi_class': ['auto'], 'n_jobs': [-1] } model = LogisticRegression() grid_search = GridSearchCV(model, [parameters, param grid_search.fit(x_train, y_train) best_parameters = grid_search.best_params_ best_score = grid_search.best_score_ print(f"The Best Parameters: {best_parameters}") print(f"Best Accuracy: {best_score}") The Best Parameters: {'C': 1.0, 'fit_intercept': True Best Accuracy: 0.7888452474469756 C:\Users\vinu0\anaconda3\Lib\site-packages\sklearn\li</pre>	e, 'max_iter': 100, 'multi_class': 'auto', 'n_jobs': -1, 'penalty': '12', 'solver': 'lbfgs', 'tol': 0.001}
<pre>'multi_class': ['auto'], 'n_jobs': [-1] } parameters_liblinear = { 'penalty': ['ll', 'l2'], 'dual': [False], 'tol': [le-4, le-3], 'C': [0.01, 0.1, 1.0], 'fit_intercept': [True, False], 'solver': ['liblinear'], 'max_iter': [100, 200], 'multi_class': ['auto'], 'n_jobs': [-1] } model = LogisticRegression() grid_search = GridSearchCV(model, [parameters, param grid_search.fit(x_train, y_train) best_parameters = grid_search.best_params_ best_score = grid_search.best_score_ print(f"The Best Parameters: {best_parameters}") print(f"Best Accuracy: {best_score}") The Best Parameters: {'C': 1.0, 'fit_intercept': True Best Accuracy: 0.7888452474469756 C:\Users\vinu0\anaconda3\Lib\site-packages\sklearn\li efault value to avoid this warning. warnings.warn(model = LogisticRegression(C=best_parameters['c'], fit_intercept=best_parameters['fit_intercept'], max_iter=best_parameters['max_iter'], multi_class=best_parameters['multi_class'],</pre>	e, 'max_iter': 100, 'multi_class': 'auto', 'n_jobs': -1, 'penalty': '12', 'solver': 'lbfgs', 'tol': 0.001}
<pre>'multi_class': ['auto'], 'n_jobs': [-1] } parameters_liblinear = { 'penalty': ['11', '12'], 'dual': [False], 'tol': [le-4, le-3], 'C': [0.01, 0.1, 1.0], 'fit_intercept': [True, False], 'solver': ['liblinear'], 'max_iter': [100, 200], 'multi_class': ['auto'], 'n_jobs': [-1] } model = LogisticRegression() grid_search = GridSearchCV(model, [parameters, param grid_search.fit(x_train, y_train) best_parameters = grid_search.best_params_ best_score = grid_search.best_score_ print(f"The Best Parameters: {best_parameters}") print(f"Best Accuracy: {best_score}") The Best Parameters: {'C': 1.0, ''fit_intercept': True Best Accuracy: 0.7888452474469756 C:\Users\vinu0\anaconda3\Lib\site-packages\sklearn\liefault value to avoid this warning. warnings.warn(: model = LogisticRegression(C=best_parameters['C'], fit_intercept=best_parameters['fit_intercept'], max_iter=best_parameters['max_iter'], multi_class=best_parameters['max_iter'], n_jobs=best_parameters['penalty'], solver=best_parameters['penalty'], solver=best_parameters['penalty'], solver=best_parameters['solver'], tol=best_parameters['tol']) model.fit(x_train, y_train) y_pred = model.predict(x_test) accuracy = accuracy_score(y_test, y_pred) print(f"Accuracy: {accuracy}")</pre>	e, 'max_iter': 100, 'multi_class': 'auto', 'n_jobs': -1, 'penalty': '12', 'solver': 'lbfgs', 'tol': 0.001}
<pre>'multi_class': ['auto'], 'n_jobs': [-1] } parameters_liblinear = { 'penalty': ['11', '12'], 'dual': [False], 'tol': [1e-4, 1e-3], 'C': [0.01, 0.1, 1.0], 'fit_intercept': [True, False], 'solver': ['liblinear'], 'max_iter': [100, 200], 'multi_class': ['auto'], 'n_jobs': [-1] } model = LogisticRegression() grid_search = GridSearchCV(model, [parameters, param grid_search.fit(x_train, y_train) best_parameters = grid_search.best_params_ best_score = grid_search.best_score_ print(f"The Best Parameters: {best_parameters}") print(f"Best Accuracy: (best_score)") The Best Parameters: {'C': 1.0, 'fit_intercept': True Best Accuracy: 0.7888452474469756 C:\Users\vinu0\anaconda3\Lib\site-packages\sklearn\li efault value to avoid this warning. warnings.warn(: model = LogisticRegression(C=best_parameters['C'], fit_intercept=best_parameters['multi_class'], n_jobs=best_parameters['multi_class'], n_jobs=best_parameters['multi_class'], n_jobs=best_parameters['penalty'], solver=best_parameters['penalty'], solver=best_parameters['solver'], tol=best_parameters['tol']) model.fit(x_train, y_train) y_pred = model.predict(x_test) accuracy = accuracy_score(y_test, y_pred) print(f"Accuracy: {accuracy}") check() C:\Users\vinu0\anaconda3\Lib\site-packages\sklearn\li efault value to avoid this warning. warnings.warn(Accuracy: 0.820627802690583 Enter input values for the given features: [0]</pre>	s, 'max_iter': 100, 'multi_class': 'auto', 'n_jobs': -1, 'penalty': '12', 'solver': 'lbfgs', 'tol': 0.001) Inear_model_logistic.py:1247: FutureWarning: 'multi_class' was deprecated in version 1.5 and will be removed in 1.7. From them on, it will always use 'multinomial'. Leave
<pre>'multi_class': ['auto'], 'n_jobs': [-1] } parameters_liblinear = { 'penalty': ['11', '12'], 'dual': [False], 'tol': [1e-4, 1e-3], 'C': [0.01, 0.1, 1.0], 'fit_intercept': [True, False], 'solver': ['liblinear'], 'max_iter': [100, 200], 'multi_class': ['auto'], 'n_jobs': [-1] } model = LogisticRegression() grid_search = GridSearchCV(model, [parameters, param grid_search.fit(x_train, y_train) best_parameters = grid_search.best_params_ best_score = grid_search.best_score_ print(f"The Best Parameters: {best_parameters}") print(f"Best Accuracy: {best_score}") The Best Parameters: {'C': 1.0, 'fit_intercept': True Best Accuracy: 0.7888452474469756 C:\Users\vinu0\anaconda3\Lib\site-packages\sklearn\li efault value to avoid this warning. warnings.warn(: model = LogisticRegression(C=best_parameters['c'], fit_intercept=best_parameters['max_iter'], multi_class=best_parameters['max_iter'], multi_class=best_parameters['max_iter'], penalty=best_parameters['penalty'], solver=best_parameters['penalty'], solver=best_parameters['solver'], tol=best_parameters['tol']) model.fit(x_train, y_train) y_pred = model.predict(x_test) accuracy = accuracy_score(y_test, y_pred) print(f"Accuracy: {accuracy}") check() C:\Users\vinu0\anaconda3\Lib\site-packages\sklearn\li efault value to avoid this warning. warnings.warn(Accuracy: 0.820627802690583 Enter input values for the given features:</pre>	wear_model_logistic.py:1247: PutureKarning: 'multi_class' was deprecated in version 1.5 and will be removed in 1.7. From them on, it will always use 'multimonial'. Leave 'multimonial' be removed in 1.7. From them on, it will always use 'multimonial'. Leave 'multimonial' be removed in 1.7. From them on, it will always use 'multimonial'. Leave 'multimonial' be removed in 1.7. From them on, it will always use 'multimonial'. Leave 'multimonial' be removed in 1.7. From them on, it will always use 'multimonial'. Leave
<pre>'multi_class': ['auto'], 'n_jobs': [-1] } parameters_liblinear = { 'penalty': ['11', '12'], 'dual': [False], 'tol': [le-4, le-3], 'c': [0.01, 0.1, 1.0], 'fit_intercept': [True, False], 'solver': ['liblinear'], 'max_iter': [100, 200], 'multi_class': ['auto'], 'n_jobs': [-1] } model = LogisticRegression() grid_search = GridSearchCV(model, [parameters, param grid_search,fit(X_train, y_train) best_parameters = grid_search.best_parameters}) print(f"The Best Parameters: (best_parameters)") print(f"Best Accuracy: (best_score_") The Best Parameters: {'C': 1.0, 'fit_intercept': True Best Accuracy: 0.788452474469756 C:\Users\vinu0\anaconda3\Lib\site-packages\sklearn\liefault value to avoid this warning. warnings.warn(model = LogisticRegression(Cebest_parameters('C'), fit_intercept=best_parameters('fit_intercept'), max_iter=best_parameters('max_iter'), multi_class=best_parameters('max_iter'), multi_class=best_parameters('penalty'), solve=best_parameters('solver'), tol=best_parameters('tol')) model.fit(x_train, y_train) y_pred = model.predict(x_test) accuracy = accuracy_score(y_test, y_pred) print(f"Accuracy: {accuracy}") check() C:\Users\vinu0\anaconda3\Lib\site-packages\sklearn\liefault value to avoid this warning. warnings.warn(Accuracy: 0.820627802690583 Enter input values for the given features: [0] passenger Class = 3 Age = 56 Count_of_siblings_and_spouse = 0 count_of_sparents_and_children = 2 Fare = 11.0 Male = 0 Embarked_from_Queenstown = 1 Embarked_from_Southampton = 0 Based on the values given, the Passenger has Unfortun C:\Users\vinu0\anaconda3\Lib\site-packages\sklearn\ba warnings.warn(Now we have completed the Linear Models by Using the Linear Models by Using the Linear Models of Using</pre>	by 'max_decs': 110, 'multi_plass': 'auto', 'm_jobs': -l, 'pensity': '12', 'eclver': 'lbdgs', 'tol': 0.011) mear_model_logistic.py:12f7: SubureVarning: 'multi_plass' was deprended in version 1.5 and will be removed in 1.7. From then on, it will always use 'multimomial', Leave mear_model_logistic.py:12f7: SubureVarning: 'multi_plass' was deprended in version 1.5 and will be removed in 1.7. From then on, it will always use 'multimomial', Leave mear_model_logistic.py:12f7: SubureVarning: 'multi_plass' was deprended in version 1.5 and will be removed in 1.7. From then on, it will always use 'multimomial', Leave mear_model_logistic.py:12f7: SubureVarning: 'multi_plass' was deprended in version 1.5 and will be removed in 1.7. From then on, it will always use 'multimomial', Leave mear_model_logistic.py:12f7: SubureVarning: 'multi_plass' was deprended in version 1.5 and will be removed in 1.7. From then on, it will always use 'multimomial', Leave mear_model_logistic.py:12f7: SubureVarning: 'multi_plass' was deprended in version 1.5 and will be removed in 1.7. From then on, it will always use 'multimomial', Leave mear_model_logistic.py:12f7: SubureVarning: 'multi_plass' was deprended in version 1.5 and will be removed in 1.7. From then on, it will always use 'multimomial', Leave mear_model_logistic.py:12f7: SubureVarning: 'multi_plass' was deprended in version 1.5 and will be removed in 1.7. From then on, it will always use 'multimomial', Leave mear_model_logistic.py:12f7: SubureVarning: 'multi_plass' was deprended in version 1.5 and will be removed in 1.7. From then on, it will always use 'multimomial', Leave mear_model_logistic.py:12f7: SubureVarning: 'multi_plass' was deprended in version 1.5 and will be removed in 1.7. From then on, it will always use 'multimomial', Leave mear_model_logistic.py:12f7: SubureVarning: 'multi_plass' was deprended in version 1.5 and will be removed in 1.7. From then on, it will always use 'multimomial', Leave mear_model_logistic.py:12f7: SubureVarning: 'multi_plass
<pre>'multi_class': ('auto'], 'n_jobs': [-1] } parameters_liblinear = { 'penalty!: ['11', '12'], 'dual': [False], 'tol': (le-4, le-3), 'C': (0.01, 0.1, 1.0), 'fit_intercept': [True, False], 'solver': ['liblinear'], 'max_iter': [100, 200], 'multi_class': ('auto'], 'n_jobs': [-1] } model = LogisticRegression() grid_search = GridSearchCv[model, [parameters, param grid_search.fit(x_train, y_train) best_parameters = grid_search.best_params_ best_score = grid_search.best_score_ print(f"The Best Parameters: ('C': 1.0, 'fit_intercept': True Best Accuracy: (best_score)") The Best Parameters: ('C': 1.0, 'fit_intercept': True Best Accuracy: (0.7888452474469756 C:\Users\vinu\anaconda3\Lib\site-packages\sklearn\liefault value to avoid this warning. warnings.warn(model = LogisticRegression(C=best_parameters('C'), fit_intercept=best_parameters('max_iter'), max_iter=best_parameters('max_iter'), max_iter=best_parameters('max_iter'), n_jobs=best_parameters('penalty'), solver=best_parameters('penalty'), solver=best_parameters('penalty'), solver=best_parameters('solver'), tol=best_parameters('test) accuracy = accuracy_score(y_test, y_pred) print(f"Accuracy: (accuracy)") check() C:\Users\vinu\anaconda3\Lib\site-packages\sklearn\liefault value to avoid this warning. warnings.warn(Accuracy: 0.820627802690583 Enter input values for the given features: [0] Passenger Class = 3 Age = 56 Count_of_siblings_and_spouse = 0 count_of_parents_and_children = 2 Fare = 111.0 Male = 0 Embarked_from_Southampton = 0 Based on the values given, the Passenger has Unfortun C:\Users\vinu\anaconda3\Lib\site-packages\sklearn\ba warnings.warn(Now we have completed the Linear Models by Using the Linear Now let us move to the Classifiers. let us take the RandomTree(parameters=('n_estimators': [50, 100], 'criterion': ['gini', 'entropy'], 'max_features': [70, 100], 'min_samples_leaf': [1, 2], 'max_features': [70, 100], 'criterion': [</pre>	by 'max_decs': 100, 'multi_pleas': 'auto', 'm_jobs': -l, 'pensity': '12', 'solver': 'lbfgs', 'tob': 0.001) meas_model_logistic.py:12f7: SubureWarning: 'multi_pleas' was deprenated in version 1.5 and will be removed in 1.7. Sizes then on, it will always use 'multimomis'', Leave meas_model_logistic.py:12f7: SubureWarning: 'multi_pleas' was deprenated in version 1.5 and will be removed in 1.7. Sizes then on, it will always use 'multimomis'', Leave meas_model_logistic.py:12f7: SubureWarning: 'multi_pleas' was deprenated in version 1.5 and will be removed in 1.7. Sizes then on, it will always use 'multimomis'', Leave meas_model_logistic.py:12f7: SubureWarning: 'multi_pleas' was deprenated in version 1.5 and will be removed in 1.7. Sizes then on, it will always use 'multimomis'', Leave meas_model_logistic.py:12f7: SubureWarning: 'multi_pleas' was deprenated in version 1.5 and will be removed in 1.7. Sizes then on, it will always use 'multimomis'', Leave meas_model_logistic.py:12f7: SubureWarning: 'multi_pleas' was deprenated in version 1.5 and will be removed in 1.7. Sizes then on, it will always use 'multimomis'', Leave meas_model_logistic.py:12f7: SubureWarning: 'multi_pleas' was deprenated in version 1.5 and will be removed in 1.7. Sizes then on, it will always use 'multimomis'', Leave meas_model_logistic.py:12f7: SubureWarning: 'multi_pleas' was deprenated in version 1.5 and will be removed in 1.7. Sizes then on, it will always use 'multimomis'', Leave meas_model_logistic.py:12f7: SubureWarning: 'multi_pleas' was deprenated in version 1.5 and will be removed in 1.7. Sizes then on, it will always use 'multimomis'', Leave meas_model_logistic.py:12f7: SubureWarning: 'multi_pleas' was deprenated in version 1.5 and will be removed in 1.7. Sizes then on, it will always use 'multimomis'', Leave meas_model_logistic.py:12f7: SubureWarning: 'multi_pleas' was deprenated in version 1.5 and will be removed in 1.7. Sizes then on, it will always use 'multimomis'', Leave meas_model_logistic.py:12f7: Subur
<pre>'multi_class': ['auto'], 'n_jobs': [-1] } parameters_liblinear = { 'penalty': ['11', '12'], 'dual': [False], 'col': [0.01, 0.1, 1.0], 'fit_intercept': [True, False], 'solver': ['liblinear'], 'max_iter': [100, 200], 'multi_class': ['auto'], 'n_jobs': [-1] } model = LogisticRegression() grid_search = GridSearchCV(model, [parameters, param grid_search = GridSearchCv(model, parameters, par</pre>	we instance: LNC, 'musticlase': 'mustic injust' injust' was degreeated in version 1.0 and will be recoved in 1.7, from them on, it will always use 'multi-contail', inseed any section 1.0 and will be recoved in 1.7, from them on, it will always use 'multi-contail', inseed any section 1.0 and will be recoved in 1.7. From them on, it will always use 'multi-contail', inseed any section 1.0 and will be recoved in 1.7. From them on, it will always use 'multi-contail', inseed any section 1.0 and will be recoved in 1.7. From them on, it will always use 'multi-contail', inseed any section 1.0 and will be recoved in 1.7. From them on, it will always use 'multi-contail', inseed any section 1.0 and will be recoved in 1.7. From them on, it will always use 'multi-contail', inseed any section 1.7. From them on, it will always use 'multi-contail', inseed any section 1.7. From them on, it will always use 'multi-contail', inseed any section 1.7. From them on, it will always use 'multi-contail', inseed any section 1.7. From them on, it will always use 'multi-contail', inseed any section 1.7. From them on, it will always use 'multi-contail', inseed any section 1.7. From them on, it will always use 'multi-contail', inseed any section 1.7. From them on, it will always use 'multi-contail', inseed any section 1.7. From them on, it will always use 'multi-contail', inseed any section 1.7. From them on, it will always use 'multi-contail', inseed any section 1.7. From them on, it will always use 'multi-contail', inseed any section 1.7. From them on, it will always use 'multi-contail', inseed any section 1.7. From them on, it will always use 'multi-contail', it will always use 'multi-co
<pre>'multi_clase': ['auto'], 'n_jobs': [-1] } parameters_liblinear = { 'penalty': ['11', '12'], 'doual': [False], 'tol': [lo-4, le-3], 'c': [0.01, 0.1, 1.0], 'fif_inercept': [True, False], 'solver': ['liblinear'], 'ma_tter': [100, 200], 'multi_clase': ['auto'], 'n_jobs': [-1] } model = LogisticRegression() grid_search = GridSearchCv(model, (parameters, param grid_search = GridSearchCv(model, (parameters, param grid_search = GridSearch.best_parameters) best_parameters = grid_search.best_parameters) best_parameters = grid_search.best_parameters) print(f'The Beat Parameters: ('C': 1.0, 'fit_intercept': True Beat Accuracy: 0.7888452474469756 C'\UseralvinuO\anaconda3\Lib\ait=packages\sklearn\liefault value to avoid this warning. warnings.warn(model = LogisticRegression(Cebest_parameters['C': 1, obs'], ma_iter=best_parameters['malti_class'], n_jobs=best_parameters['malti_class'], n_jobs=best_parameters['malti_class'], n_jobs=best_parameters['malti_class'], n_jobs=best_parameters['malti_class'], n_jobs=best_parameters['malti_class'], n_jobs=best_parameters['tolver'], tol=best_parameters['tolver'], tol=best_parameters['tol=best_parameters] parameters = grid_search_best_parameters, ov = 5,s grid_search_best_parameters, ov = 5,s grid_searc</pre>	we instance: LNC, 'musticlase': 'mustic injust' injust' was degreeated in version 1.0 and will be recoved in 1.7, from them on, it will always use 'multi-contail', inseed any section 1.0 and will be recoved in 1.7, from them on, it will always use 'multi-contail', inseed any section 1.0 and will be recoved in 1.7. From them on, it will always use 'multi-contail', inseed any section 1.0 and will be recoved in 1.7. From them on, it will always use 'multi-contail', inseed any section 1.0 and will be recoved in 1.7. From them on, it will always use 'multi-contail', inseed any section 1.0 and will be recoved in 1.7. From them on, it will always use 'multi-contail', inseed any section 1.0 and will be recoved in 1.7. From them on, it will always use 'multi-contail', inseed any section 1.7. From them on, it will always use 'multi-contail', inseed any section 1.7. From them on, it will always use 'multi-contail', inseed any section 1.7. From them on, it will always use 'multi-contail', inseed any section 1.7. From them on, it will always use 'multi-contail', inseed any section 1.7. From them on, it will always use 'multi-contail', inseed any section 1.7. From them on, it will always use 'multi-contail', inseed any section 1.7. From them on, it will always use 'multi-contail', inseed any section 1.7. From them on, it will always use 'multi-contail', inseed any section 1.7. From them on, it will always use 'multi-contail', inseed any section 1.7. From them on, it will always use 'multi-contail', inseed any section 1.7. From them on, it will always use 'multi-contail', inseed any section 1.7. From them on, it will always use 'multi-contail', it will always use 'multi-co
<pre>"multi_class": ['auto'], 'n_jobs': [-1] } parameters_liblinear = { 'penalty": ['11', '12'], 'dual': [False], 'co': [0.01, 0.1, 1.0], 'fit_intercept': [True, False], 'solver': ['liblinear'], 'max_ister': [100, 200], 'multi_class': ['auto'], 'n_jobs': [-1] } model = LogisticRegression() grid_search = GridSearch(V(model, [parameters, param grid_search() = Grid_search(V(model, parameters, parameters)] print(f*Nest_Accuracy: (best_score)) The Best Parameters: {C': 1.0, 'fit_intercept': True Best Accuracy: 0.7888452474469756 C:\Users\vinu0\anaconds3\Lib\site-packages\sklearn\liefault value to avoid this warning. warnings.warn(cobest_parameters('c'), model.fit(max_interparameters('parameters('nulti_class'), n_jobs-best_parameters('parameters('max_iter'), mold.fit(max_interparameters('ro)) solver=best_parameters('ro) solver=best</pre>	A conclusion 120. Smallest construction of the
<pre>"milobs: [-1] } parameters_liblinear = { 'penalty!: ['ll', 'l2'], 'dual': [Paise], 'tol': [le-4, le-3], 'ct: [0.01, 0.1, 1.0], 'fit_intercept: [True, Faise], 'salv_ite': ['liblinear'], 'max_ite': ['liblinear'], 'max_ite': ['liblinear'], 'max_ite': ['liblinear'], 'max_ite': ['liblinear'], 'max_ite': [100, 200], 'malti_class': ['auto'], 'l_jobs': [-1] } model = LogisticRegression() grid.search = Grid.search best_parameters, param grid.search.fir(x_train, y_train) best_parameters = grid.search best_parameters,") print(f'Best_Accuracy: (best_score)") The Sest_Parameters: ['C': 1.0, 'fit_intercept': True Best_Accuracy: 0.78884247469756 C'\Users\vinu0\anaccnda2\tib\aits_packages\sklearn\liefault value to avoid this warning, warnings_warn model = LogisticRegression(Cheeks_parameters('C'), fit_intercept*best_parameters('fit_intercept'), max_iter*best_parameters('fit_intercept'), max_iter*best_parameters('fit_intercept'), model.parameters('parameters('fit_intercept'), model.parameters('parameters('parameters'), penalty*best_parameters('fit_intercept'), model.fit(X_train, y_train) y_pred = model.predict(X_test) accuracy = acctracy_score(y_test, y_pred) print(f'Accuracy: (accuracy)') hotel, value to avoid this warning. varnings_warn C'\Users\vinu0\anaccnda3\Lib\site_packages\sklearn\liefault value to avoid this warning. varnings.warn C\users\vinu0\anaccnda3\Lib\site_packages\sklearn\liefault', value for the given features: [O\users\vinu0\anaccnda3\Lib\site_packages\sklearn\liefault', value for the given features: [O\users\vin</pre>	the process of the filter of the filter of the control of the cont
<pre>"mailing.lamaers ("autor"), 'm.jobs': [-1] } parameters_liblinesr = ('penalty': ['ll', 'll'), 'dual': [Peales], 'tol': [le-4, le-3], 'c': [0.0., 0.1. 1.0], 'fit_intercopt': ['True. False], 'easiver': ['lliblinear'], 'max_iter': [[10.0. 200], 'mmid_lease': ['Putor'], 'molel = LogisticRegression() 'grid_wearch = StidSearchOv(ondel, [parameters, paramyrid, search = StidSearchOv(ondel, [parameters, paramyrid, search = StidSearchOv(ondel, [parameters, paramyrid, search = StidSearchOv(ondel, [parameters, paramyrid, search.fit(X,train, y,train) bet_parameters = grid_pearch.best_parameters]' print(f'Febet Accuracy: [sea_parameters] bet_parameters = grid_pearch.best_parameters]' print(f'Febet Accuracy: [sea_parameters]' print(f'Febet Accuracy: [sea_parameters]' c'slueers'windNanaeconds'ulb'ste-packages\skiearn\sietault' value to avoid this warning. warning.warn(chest_parameters['C':].0. 'fit_intercept'], salice=beat_parameters['max_iter'], sulic_lolamsbebt_parameters['penalty'], solve=beat_parameters['cho']) incidel = logisticRegression(chest_parameters['cho']) panalty=beat_parameters['cho']) incidel.fit(X_train, y_train) y_pred = model.predict(X_tbat) parameters = grid_parameters['tol'] } parameters = grid_parameters['tol'] } colled = socuracy_parameters['tol'] } colled = socuracy_parameters['tol'] } colled = fit(X_train, y_train) y_pred = model.predict(X_tbat) parameters parameters = grid_parameters = grid_parameters 'n_socimeters' = grid_parameters 'n_socimeters' = grid_parameters' 'n_socimeters' = grid_parameters' = grid_parameters' 'n_socimeters' = grid_parameters' = grid_parameters' 'n_socimeters' = grid_parameters' = grid</pre>	the property of the property o
<pre>"multiclases: ('auto'), 'n_jounity': ['11', '12'], 'deal': [Raise], 'col': [Indameropt': [True, False], 'solver': ('Iliolnamer'), 'makitar': [100, 200], 'makitar': [100, 200], 'makitar': ['100, 200], 'makitar': ['100,</pre>	Consideration of the contest of the
'mujclesons' ('auton'), 'mujcles' ('all') 'mujcles' ('all'), 'toul's [les's les'), 'toul's [les's les'), 'toul's [les's les'), 'four intercept's ['free, Paise], 'solver'] ('labolinear'), 'most tert's [100, 200], 'multicless's ('auton'), 'mujcless's ('auton'), 'mujcless	AND AND ADDRESS OF THE ADDRESS OF TH
'moles':[-11] promotern_liblinear = {	AND AND ADDRESS OF THE PROPERTY OF THE LOCATION OF THE PROPERTY OF THE PROPERY OF THE PROPERTY OF THE PROPERTY OF THE PROPERTY OF THE PROPERTY
multicolase's [Paucol], multicolase's [10] parameters_low(ness : param	AND

Therefore I have built a reliable predictive model for passenger survival, emphasizing key features like class, age, and fare. The project highlighted the power of data-driven analysis in historical contexts.

Survival Prediction on the Titanic Dataset